

Ability to Count Is Worth $\Theta(\Delta)$ Rounds

Tuomo Lempinen

Helsinki Institute for Information Technology HIIT & Department of Computer Science,
Aalto University, Finland

Abstract

Hella et al. (PODC 2012, Distributed Computing 2015) identified seven different models of distributed computing—one of which is the port-numbering model—and provided a complete classification of their computational power relative to each other. However, one of their simulation results involves an additive overhead of $2\Delta - 2$ communication rounds, and it was not clear, if this is actually optimal. In this paper we give a positive answer: there is a matching linear-in- Δ lower bound. This closes the final gap in our understanding of the models, with respect to the number of communication rounds.

1 Introduction

This work studies the significance of being able to count the multiplicities of identical incoming messages in distributed algorithms. We compare two models: one, in which each node receives a *set* of messages in each round, and another, in which each node receives a *multiset* of messages in each round. It has been previously shown that the latter model can be simulated in the former model by allowing an additive overhead of linear in Δ communication rounds, where Δ is the maximum degree of the graph [8]. In this work we show that this is optimal: in some cases, linear in Δ extra rounds are strictly necessary.

1.1 A Hierarchy of Weak Models

The models that we study are weaker variants of the well-known *port-numbering model*. Hella et al. [8] defined a collection of seven models, one of which is the port-numbering model. We denote by VV_c the class of all *graph problems* that can be solved in this model. The following subclasses of VV_c correspond to the weaker variants:

- VV: Input and output ports connected to the same neighbour do not necessarily have the same number.
- MV: Input ports are not numbered; nodes receive a multiset of messages.
- SV: Input ports are not numbered; nodes receive a set of messages.
- VB: Output ports are not numbered; nodes broadcast the same message to all neighbours.
- MB: Combination of MV and VB.
- SB: Combination of SV and VB.

There are some trivial containment relations between the classes, such as $SV \subseteq MV \subseteq VV \subseteq VV_c$. The trivial relations are depicted in Figure 1a. However, some classes, such as VB and SV, are seemingly orthogonal. Somewhat surprisingly, Hella et al. [8] were able to show that the classes form a linear order:

$$SB \subsetneq MB = VB \subsetneq SV = MV = VV \subsetneq VV_c.$$

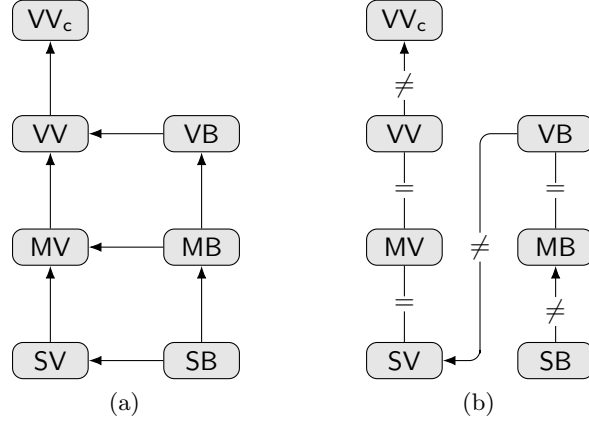


Figure 1: (a) Trivial containment relations between the problem classes. (b) The linear order obtained by Hella et al. [8].

For each class, we can also define the subclass of problems solvable in constant time independent on the size of the input graph. The same containment relations hold for the constant-time versions of the classes. The relations are depicted in Figure 1b.

The equalities between classes are proved by showing that algorithms corresponding to a seemingly more powerful class can be simulated by algorithms corresponding to a seemingly weaker class. In the case of $SV = MV$, there is an overhead involved, whereas the rest of the simulation results do not increase the running time.

1.2 Classes SV and MV

In this work we study further the relationship between the models that are related to the classes SV and MV . Neither of the models features incoming port numbers. The only difference is that in the case of MV , algorithms are able to count the number of neighbours that sent any particular message, while in SV this is not possible. For now we will use informally the terms *SV-algorithm* and *MV-algorithm*; a more formal definition will follow in Section 2.

Hella et al. [8] proved that any MV -algorithm can be simulated by an SV -algorithm, given that the simulating algorithm is allowed to use $2\Delta - 2$ extra communication rounds. The basic idea is that when nodes gather all available information from their radius- $(2\Delta - 2)$ neighbourhood, the outgoing port numbers necessarily break symmetry. Any neighbours u and w of a node v either have different outgoing port numbers towards v or are in a different internal state. This symmetry breaking information can then be used during the simulation to receive a distinct message from each neighbour.

1.3 Contributions

This work gives tight lower bounds for simulating MV -algorithms by SV -algorithms. We will prove two theorems. The first theorem is about a so-called simulation problem, that is, breaking symmetry between incoming messages. It is intended to be an exact counterpart to the upper bound result given by the simulation algorithm of Hella et al. [8].

Theorem 1. *For each $\Delta \geq 2$ there is a port-numbered graph of maximum degree Δ with nodes v, u, w , such that when executing any SV -algorithm in the graph, node v receives identical messages from its neighbours u and w in rounds $1, 2, \dots, 2\Delta - 2$.*

Our second theorem gives a *graph problem* that separates MV -algorithms from SV -algorithms with respect to running time as a function of the maximum degree Δ .

Theorem 2. *There is a graph problem that can be solved in one communication round by an MV-algorithm, but that requires at least Δ rounds for all $\Delta \geq 2$, when solved by an SV-algorithm.*

Our results are based on constructing a family of graphs with an intricate port numbering of certain kind. We start by proving Theorem 1 in Section 3, and then we adapt the same construction to prove Theorem 2 in Section 4.

1.4 Motivation and Related Work

The port-numbering model, or VV_c , can be thought to model wired networks, whereas model SB corresponds to fully wireless systems. Other models in the hierarchy are intermediate steps between the two extreme cases.

Models similar to MV have been studied previously under various names: output port awareness [4], wireless in input [5], mailbox [5], port-to-mailbox [11] and port-à-boîte [6]. However, most of the previous research does not give general results about graph problems, but instead focuses on individual problems or makes different assumptions about the model. To the best of our knowledge, model SV has not been studied before the work of Hella et al. [8].

Emek and Wattenhofer [7] have considered networks of nodes with very limited computation and communication capabilities—in particular, the nodes can count identical messages only up to some predetermined number. They argue that models like that will be crucial when applying distributed computing to networks of biological cells.

Our models have analogies also in graph exploration. Models SV and MV correspond to the case where an agent does not know from which edge it arrived to a node. This is true for *traversal sequences* [1], as opposed to *exploration sequences* [9]. If we have several agents exploring a graph, the question of whether they can count the number of identical agents in a node becomes interesting. Our lower bounds indicate that, with appropriate definitions, this ability causes a difference of linear in Δ steps in certain traversal sequences.

Hella et al. [8] identified a connection between the seven models of computation and certain variants of modal logic, in the spirit of descriptive complexity theory. In certain classes of structures, *graded multimodal logic* corresponds to MV and *multimodal logic* corresponds to SV . Thus our lower-bound result can be recast in terms of modal formulas: when given a formula ϕ of graded multimodal logic, we can find an equivalent formula ψ of multimodal logic, but in general, the modal depth $\text{md}(\psi)$ of ψ has to be at least $\text{md}(\phi) + \Delta - 1$. For details on modal logic, see Blackburn, Rijke and Venema [3] or Blackburn, Benthem and Wolter [2].

2 Preliminaries

In this section we define the models of computation and the problems we study, as well as introduce tools that will be needed in order to prove our results.

2.1 Distributed Algorithms

We define distributed algorithms as state machines. They are executed in a graph such that each node of the graph is a copy of the same state machine. Nodes can communicate with adjacent nodes. In this work, we consider only deterministic state machines and synchronous communication in anonymous networks.

In the beginning of execution, each state machine is initialised based on the degree of the node and a possible local input given to it. Then, in each communication round, each state machine performs three operations:

- (1) sends a message to each neighbour,
- (2) receives a message from each neighbour,

(3) moves to a new state based on the current state and the received messages.

If the new state is a special stopping state, the machine halts. The local output of the node is its state after halting. Next, we will define distributed systems more formally.

2.1.1 Inputs and Port Numberings

Consider a graph $G = (V, E)$. An *input* for G is a function $f: V \rightarrow X$, where X is a finite set such that $\emptyset \in X$. For each $v \in V$, the value $f(v)$ is called the *local input* of v .

A *port* of G is a pair (v, i) , where $v \in V$ is a node and $i \in [\deg(v)]$ is the number of the port. Let $P(G)$ be the set of all ports of G . A *port numbering* of G is a bijection $p: P(G) \rightarrow P(G)$ such that

$$p(v, i) = (u, j) \text{ for some } i \text{ and } j \text{ if and only if } \{v, u\} \in E.$$

Intuitively, if $p(v, i) = (u, j)$, then (v, i) is an output port of node v that is connected to an input port (u, j) of node u .

When analysing lower-bound constructions, we will find the following generalisation of port numbers useful. Let N be an arbitrary set. Assume that for each $v \in V$, $I_v \subseteq N$ and $O_v \subseteq N$ are subsets of size $\deg(v)$. Now, a *generalised input port* is a pair (v, i) , where $v \in V$ and $i \in I_v$, and a *generalised output port* is a pair (v, o) , where $v \in V$ and $o \in O_v$. A *generalised port numbering* p is then a bijection that maps each output port to an input port of an adjacent node.

2.1.2 State Machines

For each positive integer Δ , denote by $\mathcal{F}(\Delta)$ the class of all simple undirected graphs of maximum degree at most Δ . Let $X \ni \emptyset$ be a finite set of local inputs. A *distributed state machine* for $(\mathcal{F}(\Delta), X)$ is a tuple $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$, where

- Y is a set of states,
- $Z \subseteq Y$ is a finite set of stopping states,
- $\sigma_0: \{0, 1, \dots, \Delta\} \times X \rightarrow Y$ is a function that defines the initial state,
- M is a set of messages such that $\epsilon \in M$,
- $\mu: Y \times [\Delta] \rightarrow M$ is a function that constructs the outgoing messages, such that $\mu(z, i) = \epsilon$ for all $z \in Z$ and $i \in [\Delta]$,
- $\sigma: Y \times M^\Delta \rightarrow Y$ is a function that defines the state transitions, such that $\sigma(z, \bar{m}) = z$ for all $z \in Z$ and $\bar{m} \in M^\Delta$.

The special symbol $\epsilon \in M$ indicates “no message” and \emptyset indicates “no input”.

2.1.3 Executions

Let $G = (V, E) \in \mathcal{F}(\Delta)$ be a graph, let p be a port numbering of G , let $f: V \rightarrow X$ be an input for G , and let \mathcal{A} be a distributed state machine for $(\mathcal{F}(\Delta), X)$. Then we can define the *execution* of \mathcal{A} in (G, f, p) as follows.

The state of the system in round $r \in \mathbb{N}$ is represented as a function $x_r: V \rightarrow Y$, where $x_r(v)$ is the *state* of node v in round r . To initialise the nodes, set

$$x_0(v) = \sigma_0(\deg(v), f(v)) \quad \text{for each } v \in V.$$

Then, assume that x_r is defined for some $r \in \mathbb{N}$. Let $(u, j) \in P(G)$ and $(v, i) = p(u, j)$. Now, node v receives the message

$$a_{r+1}(v, i) = \mu(x_r(u), j)$$

from its port (v, i) in round $r + 1$. For each $v \in V$, we define a vector of length Δ consisting of messages received by node v in round $r + 1$ and the symbol ϵ :

$$\bar{a}_{r+1}(v) = (a_{r+1}(v, 1), a_{r+1}(v, 2), \dots, a_{r+1}(v, \deg(v)), \epsilon, \epsilon, \dots, \epsilon),$$

where the padding with the special symbol ϵ is to simplify our notation so that $\bar{a}_{r+1}(v) \in M^\Delta$. Now we can define the new state of each node $v \in V$ as follows:

$$x_{r+1}(v) = \sigma(x_r(v), \bar{a}_{r+1}(v)).$$

Let $t \in \mathbb{N}$. If $x_t(v) \in Z$ for all $v \in V$, we say that \mathcal{A} *stops in time t* in (G, f, p) . The *running time* of \mathcal{A} in (G, f, p) is the smallest t for which this holds. If \mathcal{A} stops in time t in (G, f, p) , the *output* of \mathcal{A} in (G, f, p) is $x_t: V \rightarrow Y$. For each $v \in V$, the *local output* of v is $x_t(v)$.

We define the *execution* of \mathcal{A} in (G, p) to be the execution of \mathcal{A} in (G, f, p) , where f is the unique function $f: V \rightarrow \{\emptyset\}$.

2.1.4 Algorithm Classes

So far, we have defined only a single model of computation. However, our aim in this work is to investigate the relationships between two variants of the model. To this end, we will now introduce two different restrictions to the definition of a state machine.

Given a vector $\bar{a} = (a_1, a_2, \dots, a_\Delta) \in M^\Delta$, define

$$\begin{aligned} \text{set}(\bar{a}) &= \{a_1, a_2, \dots, a_\Delta\}, \\ \text{multiset}(\bar{a}) &= \{(m, n) : m \in M, n = |\{i \in [\Delta] : m = a_i\}|\}. \end{aligned}$$

That is, $\text{set}(\bar{a})$ discards the ordering and multiplicities of the elements of \bar{a} , while $\text{multiset}(\bar{a})$ discards only the ordering.

Now we can define classes \mathcal{SV} and \mathcal{MV} of state machines. Class \mathcal{SV} consists of all distributed state machines $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$ such that

$$\text{set}(\bar{a}) = \text{set}(\bar{b}) \quad \text{implies} \quad \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \quad \text{for all } y \in Y.$$

Similarly, class \mathcal{MV} consists of all distributed state machines $\mathcal{A} = (Y, Z, \sigma_0, M, \mu, \sigma)$ such that

$$\text{multiset}(\bar{a}) = \text{multiset}(\bar{b}) \quad \text{implies} \quad \sigma(y, \bar{a}) = \sigma(y, \bar{b}) \quad \text{for all } y \in Y.$$

The idea here is that for state machines in \mathcal{MV} , the state transitions are invariant with respect to the order of incoming messages; in practice, nodes receive the messages in a multiset. In \mathcal{SV} , nodes receive the messages in a set, which means that the state transitions are invariant with respect to both the order and multiplicities of incoming messages.

We will later find useful the following definitions for infinite sequences of state machines, where Δ will be used as an upper bound for the maximum degree of graphs:

$$\begin{aligned} \mathbf{MV} &= \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{MV} \text{ for all } \Delta\}, \\ \mathbf{SV} &= \{(\mathcal{A}_1, \mathcal{A}_2, \dots) : \mathcal{A}_\Delta \in \mathcal{SV} \text{ for all } \Delta\}. \end{aligned}$$

From now on, both distributed state machines \mathcal{A} and sequences of distributed state machines \mathbf{A} will be referred to as *algorithms*. The precise meaning should be clear from the notation.

2.2 Graph Problems

Let X and Y be finite nonempty sets. A *graph problem* is a function $\Pi_{X,Y}$ that maps each undirected simple graph $G = (V, E)$ and each input $f: V \rightarrow X$ to a set $\Pi_{X,Y}(G, f)$ of solutions. Each *solution* $S \in \Pi_{X,Y}(G, f)$ is a function $S: V \rightarrow Y$. We handle problems without local input by setting $X = \{\emptyset\}$. One can see that our definition covers a large selection of typical distributed computing problems, such as those where the task is to find a subset or colouring of vertices.

Let $\Pi_{X,Y}$ be a graph problem, $T: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ a function and $\mathbf{A} = (\mathcal{A}_1, \mathcal{A}_2, \dots)$ a sequence such that each \mathcal{A}_Δ is a distributed state machine for $(\mathcal{F}(\Delta), X)$. We define that \mathbf{A} *solves* $\Pi_{X,Y}$ *in time* T if the following conditions hold for all $\Delta \in \mathbb{N}$, all finite graphs $G = (V, E) \in \mathcal{F}(\Delta)$, all inputs $f: V \rightarrow X$ and all port numberings p of G :

- (1) \mathcal{A}_Δ stops in time $T(\Delta, |V|)$ in (G, f, p) .
- (2) The output of \mathcal{A}_Δ in (G, f, p) is in $\Pi_{X,Y}(G, f)$.

If there exists a function $T: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that \mathbf{A} solves $\Pi_{X,Y}$ in time T , we say that \mathbf{A} *solves* $\Pi_{X,Y}$ or that \mathbf{A} *is an algorithm for* $\Pi_{X,Y}$. If the value $T(\Delta, n)$ does not depend on n , that is, if we have $T(\Delta, n) = T'(\Delta)$ for some function $T': \mathbb{N} \rightarrow \mathbb{N}$, we say that \mathbf{A} *solves* $\Pi_{X,Y}$ *in constant time* or that \mathbf{A} *is a local algorithm for* $\Pi_{X,Y}$.

Remark 3. Local inputs do not add anything essential to our work. Since the set X of possible input values is uniformly finite, the information given by an input $f: V \rightarrow X$ could be encoded as topological information in the graph. However, the use of local inputs will make our life easier, when we construct problem instances in Section 4.

2.2.1 Problem Classes

Now we are ready to define complexity classes based on our different notions of algorithms. The two classes studied in this work are as follows:

- **MV** consists of problems Π such that there is an algorithm $\mathbf{A} \in \mathbf{MV}$ that solves Π .
- **SV** consists of problems Π such that there is an algorithm $\mathbf{A} \in \mathbf{SV}$ that solves Π .

For both classes, we can also define their constant-time variants:

- **MV(1)** consists of problems Π such that there is $\mathbf{A} \in \mathbf{MV}$ that solves Π in constant time.
- **SV(1)** consists of problems Π such that there is $\mathbf{A} \in \mathbf{SV}$ that solves Π in constant time.

Observe that it follows trivially from the definitions of the algorithm classes that $\mathbf{SV} \subseteq \mathbf{MV}$ and $\mathbf{SV}(1) \subseteq \mathbf{MV}(1)$. It was shown by Hella et al. [8] that we actually have $\mathbf{SV} = \mathbf{MV}$ and $\mathbf{SV}(1) = \mathbf{MV}(1)$.

2.3 Bisimulation

In this section we introduce a tool that we will need when proving lower-bound results in Sections 3 and 4. The tool in question is bisimulation, and in particular, its finite approximation, which we call r -bisimulation. Simply put, a bisimulation is a relation between two structures such that related elements have identical local information and equivalent relations to other elements. For more details on bisimulation in general, see Blackburn, Rijke and Venema [3] or Blackburn, Benthem and Wolter [2].

Hella et al. [8] demonstrated the use of bisimulation in distributed computing by establishing a connection between the weak models mentioned in Section 1.1 and certain variants of modal logic. Here we take a considerably simpler approach and show directly that bisimilarity implies indistinguishability by distributed algorithms.

The general concept of bisimulation can be adapted to take into account the different amounts of information that is available to algorithms in each model. We will need only one variant in this work, the one corresponding to the class \mathbf{SV} .

Definition 4. Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G' , respectively, and let p and p' be generalised port numberings of G and G' , respectively. We define r - \mathcal{SV} -bisimilarity recursively. As a base case, we say that nodes $v \in V$ and $v' \in V'$ are 0 - \mathcal{SV} -bisimilar if $\deg_G(v) = \deg_{G'}(v')$ and $f(v) = f'(v')$. For $r \in \mathbb{N}_+$, we say that $v \in V$ and $v' \in V'$ are r - \mathcal{SV} -bisimilar if the following conditions hold:

- (B1) Nodes v and v' are 0 - \mathcal{SV} -bisimilar.
- (B2) If $\{v, w\} \in E$, then there is $w' \in V'$ with $\{v', w'\} \in E'$ such that w and w' are $(r-1)$ - \mathcal{SV} -bisimilar, and $p(w, a) = (v, b)$ and $p'(w', a) = (v', c)$ hold for some a, b, c .
- (B3) If $\{v', w'\} \in E'$, then there is $w \in V$ with $\{v, w\} \in E$ such that w and w' are $(r-1)$ - \mathcal{SV} -bisimilar, and $p(w, a) = (v, b)$ and $p'(w', a) = (v', c)$ hold for some a, b, c .

If $v \in V$ and $v' \in V'$ are r - \mathcal{SV} -bisimilar, we write $(G, f, v, p) \leftrightarrow_r^{\mathcal{SV}} (G', f', v', p')$ —or simply $v \leftrightarrow_r^{\mathcal{SV}} v'$, if the graphs, inputs and generalised port numberings are clear from the context.

It is clear from the definition that if $(G, f, v, p) \leftrightarrow_r^{\mathcal{SV}} (G', f', v', p')$ holds for some r , then $(G, f, v, p) \leftrightarrow_t^{\mathcal{SV}} (G', f', v', p')$ holds for all $t = 0, 1, \dots, r$. As the following lemma shows, r -bisimilarity entails indistinguishability by distributed algorithms up to running time r .

Lemma 5. Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G' , respectively, and let p and p' be port numberings of G and G' , respectively. If $(G, f, v, p) \leftrightarrow_r^{\mathcal{SV}} (G', f', v', p')$ for some $r \in \mathbb{N}$, $v \in V$ and $v' \in V'$, then for all algorithms $\mathcal{A} \in \mathcal{SV}$ we have $x_t(v) = x'_t(v')$ for all $t = 0, 1, \dots, r$, that is, the states of v and v' are identical in rounds $0, 1, \dots, r$.

Proof. We prove the claim by induction on r . Let $\mathcal{A} \in \mathcal{SV}$ be an arbitrary algorithm. The base case $r = 0$ is clear: since $v \leftrightarrow_0^{\mathcal{SV}} v'$, we have

$$x_0(v) = \sigma_0(\deg(v), f(v)) = \sigma_0(\deg(v'), f'(v')) = x'_0(v').$$

Suppose then that the claim holds for $r = s$ and that $v \leftrightarrow_{s+1}^{\mathcal{SV}} v'$. We obtain immediately by the inductive hypothesis that $x_t(v) = x'_t(v')$ for all $t = 0, 1, \dots, s$. Conditions (B2) and (B3) of Definition 4 guarantee that for each neighbour u of v there is a neighbour u' of v' , and vice versa, such that $u \leftrightarrow_s^{\mathcal{SV}} u'$, and additionally, $p(u, j) = (v, i)$ and $p'(u', j) = (v', i')$ for some j, i, i' . For each such pair of neighbours, the inductive hypothesis implies that $x_s(u) = x'_s(u')$. We have now $\mu(x_s(u), j) = \mu(x'_s(u'), j)$ and thus $a_{s+1}(v, i) = a'_{s+1}(v', i')$. That is, for each message $a_{s+1}(v, k)$ in the vector $\bar{a}_{s+1}(v)$ there is an identical message $a'_{s+1}(v', k')$ in $\bar{a}'_{s+1}(v')$, and vice versa. Additionally, as $\deg(v) = \deg(v')$, the special symbol ϵ is either in both of the vectors or in neither of them. It follows that $\text{set}(\bar{a}_{s+1}(v)) = \text{set}(\bar{a}'_{s+1}(v'))$. Since $\mathcal{A} \in \mathcal{SV}$, we have

$$x_{s+1}(v) = \sigma(x_s(v), \bar{a}_{s+1}(v)) = \sigma(x'_s(v'), \bar{a}'_{s+1}(v')) = x'_{s+1}(v').$$

Now $x_t(v) = x'_t(v')$ for all $t = 0, 1, \dots, s+1$, and hence we have shown that the claim holds for $r = s+1$. \square

It is quite straightforward to show by induction that r - \mathcal{SV} -bisimilarity is an equivalence relation. Since we will only need transitivity in this work, the following lemma suffices.

Lemma 6. The r - \mathcal{SV} -bisimilarity relation $\leftrightarrow_r^{\mathcal{SV}}$ is transitive in the class of quadruples (G, f, v, p) , where $G = (V, E)$ is a graph, f is an input for G , p is a generalised port numbering of G and $v \in V$.

Proof. We proceed by induction on r . The base case $r = 0$ is clear: if $(G, f, v, p) \leftrightarrow_0^{SV} (G', f', v', p')$ and $(G', f', v', p') \leftrightarrow_0^{SV} (G'', f'', v'', p'')$, then $(G, f, v, p) \leftrightarrow_0^{SV} (G'', f'', v'', p'')$. Suppose then that relation \leftrightarrow_r^{SV} is transitive for $r = s$ and that we have $(G, f, v, p) \leftrightarrow_{s+1}^{SV} (G', f', v', p')$ and $(G', f', v', p') \leftrightarrow_{s+1}^{SV} (G'', f'', v'', p'')$. Condition (B1) for v and v'' is equivalent to the base case. If $\{v, u\} \in E$, condition (B2) for v and v' implies that there is $u' \in V'$ with $\{v', u'\} \in E'$ such that $u \leftrightarrow_s^{SV} u'$, and additionally, $p(u, j) = (v, i)$ and $p'(u', j) = (v', i')$ for some j, i, i' . Then, condition (B2) for v' and v'' implies that there is $u'' \in V''$ with $\{v'', u''\} \in E''$ such that $u' \leftrightarrow_s^{SV} u''$ and $p''(u'', j) = (v'', i'')$ for some i'' . By the inductive hypothesis, we have $u \leftrightarrow_s^{SV} u''$, and thus v and v'' satisfy condition (B2). The case of the reverse condition (B3) is very similar. We obtain $(G, f, v, p) \leftrightarrow_{s+1}^{SV} (G'', f'', v'', p'')$, which shows that \leftrightarrow_r^{SV} is transitive for $r = s + 1$. \square

Finally, when given a generalised port numbering and a bisimilarity result, we need to be able to introduce an ordinary port numbering in order to actually apply the result to distributed algorithms. The following lemma shows that we can do this.

Lemma 7. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs, let f and f' be inputs for G and G' , respectively, and let p and p' be generalised port numberings of G and G' , respectively, with port numbers taken from a set N . Suppose that q and q' are port numberings of G and G' , respectively, such that $p(v, i) = (u, j)$ implies $q(v, g(i)) = (u, g(j))$ and $p'(v, i) = (u, j)$ implies $q'(v, g(i)) = (u, g(j))$ for some function $g: N \rightarrow \mathbb{N}_+$. Then $(G, f, v, p) \leftrightarrow_r^{SV} (G', f', v', p')$ implies $(G, f, v, q) \leftrightarrow_r^{SV} (G', f', v', q')$ for all $v \in V$ and $v' \in V'$.*

Proof. We prove the claim by induction on r . The base case $r = 0$ is clear, since it does not depend on (generalised) port numbers. Suppose then that the claim holds for $r = s$ and $(G, f, v, p) \leftrightarrow_{s+1}^{SV} (G', f', v', p')$. Condition (B1) is equivalent to the base case. If $\{v, u\} \in E$, then by condition (B2) there is $u' \in V'$ with $\{v', u'\}$ such that $(G, f, u, p) \leftrightarrow_s^{SV} (G', f', u', p')$, and $p(u, j) = (v, i)$ and $p'(u', j) = (v', i')$ hold for some j, i, i' . From the inductive hypothesis we get $(G, f, u, q) \leftrightarrow_s^{SV} (G', f', u', q')$, and by assumption, $q(u, g(j)) = (v, g(i))$ and $q'(u', g(j)) = (v', g(i'))$. Hence condition (B2) holds also with respect to q and q' . The case (B3) is similar. This shows that $(G, f, v, q) \leftrightarrow_{s+1}^{SV} (G', f', v', q')$ and thus the claim holds for $r = s + 1$. \square

3 A Lower Bound for the Simulation Overhead

Let us begin by restating the result that we will prove in this section.

Theorem 1. *For each $\Delta \geq 2$ there is a graph $G = (V, E) \in \mathcal{F}(\Delta)$, a port numbering p of G and nodes $v, u, w \in V$ such that when executing any algorithm $\mathcal{A} \in \mathcal{SV}$ in (G, p) , node v receives identical messages from its neighbours u and w in rounds $1, 2, \dots, 2\Delta - 2$.*

To prove Theorem 1, we define for each $d = 2, 3, \dots$ a graph $G_d = (V_d, E_d)$ of maximum degree d . The graph itself is just a rooted tree, but it gives rise to a port numbering with certain properties. The set V_d of nodes consists of sequences of pairs (i, j) , where $i, j \in \{0, 1, \dots, d\}$ will serve as a basis for port numbers, as we will see later. The sequence can be thought as a path leading from the root to the node itself. Our fundamental idea is that we construct the graph one level of nodes at a time, starting from the root, and assign generalised port numbers to each edge of a node by choosing the smallest numbers that have not yet been taken. The choice depends slightly on whether the level in question is even or odd.

We define the set V_d of nodes recursively as follows:

$$(G1) \quad \emptyset \in V_d.$$

$$(G2) \quad ((1, 0)), ((2, 1)), ((3, 2)), ((4, 3)), \dots, ((d, d-1)) \in V_d.$$

(G3) If $(a_1, a_2, \dots, a_i) \in V_d$, where i is odd and $i < 2d$, then $(a_1, a_2, \dots, a_{i+1}^j) \in V_d$ for all $j = 1, 2, \dots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j)$ is defined as follows. Let $(b_1, b_2) = a_i$ and $b_2^+ = 1$ if $b_2 = 0$, $b_2^+ = b_2$ otherwise. Define

$$\begin{aligned} c_1^j &= \min(\{1, 2, \dots, d\} \setminus \{b_2^+, c_1^1, c_1^2, \dots, c_1^{j-1}\}), \\ c_2^j &= \min(\{1, 2, \dots, d\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}). \end{aligned}$$

(G4) If $(a_1, a_2, \dots, a_i) \in V_d$, where i is even and $0 < i < 2d$, then $(a_1, a_2, \dots, a_{i+1}^j) \in V_d$ for all $j = 1, 2, \dots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j)$ is defined as follows. Let $(b_1, b_2) = a_i$. Define

$$\begin{aligned} c_1^j &= \min(\{1, 2, \dots, d\} \setminus \{b_2, c_1^1, c_1^2, \dots, c_1^{j-1}\}), \\ c_2^j &= \min(\{0, 1, \dots, d-1\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}). \end{aligned}$$

The set E_d of edges consists of all pairs $\{v, u\}$, where $v = (a_1, a_2, \dots, a_i) \in V_d$ and $u = (a_1, a_2, \dots, a_i, a_{i+1}) \in V_d$ for some $i \in \{0, 1, \dots\}$. See Figure 2 for an illustration of the radius-3 neighbourhood of node \emptyset of G_5 .

Consider nodes $v = (a_1, a_2, \dots, a_i)$ and $u = (a_1, a_2, \dots, a_{i+1})$, where $a_{i+1} = (b_1, b_2)$. The values b_1 and b_2 serve as generalised port numbers for the edge $\{v, u\}$. We define $p_d(v, b_1) = (u, b_2)$ and $p_d(u, b_2) = (v, b_1)$. The incoming port numbers will be irrelevant in this proof, since we only consider algorithms in the classes \mathcal{SV} and \mathcal{MV} . Thus, we will mostly use the notation $\pi_d(v, u) = b_1$ and $\pi_d(u, v) = b_2$ to denote the outgoing port numbers.

If $v = (a_1, a_2, \dots, a_i)$ and $u = (a_1, a_2, \dots, a_{i+1})$, we say that node v is the *parent* of node u and that u is a *child* of v . We say that the node v is *even* if i is even and *odd* if i is odd. If $a_i = (b_1, b_2)$, we call (b_1, b_2) the *type* of node v .

A *walk* is a sequence $\bar{v} = (v_0, v_1, \dots, v_k)$ of nodes such that $\{v_i, v_{i+1}\} \in E_d$ for all $i = 0, 1, \dots, k-1$. A pair (\bar{v}_1, \bar{v}_2) of walks, where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$, and $k \leq 2d-3$, is called a *pair of compatible walks (PCW) of length k in G_d* if the following two conditions hold:

$$(W1) \quad v_0^1 = ((1, 0)) \text{ and } v_0^2 = ((2, 1)).$$

$$(W2) \quad \pi_d(v_j^1, v_{j-1}^1) = \pi_d(v_j^2, v_{j-1}^2) \text{ for all } j = 1, 2, \dots, k.$$

If we additionally have the following for a PCW, it is called a *pair of separating walks (PSW)*:

$$(W3) \quad \text{There is } v_{k+1}^1 \in V_d \text{ with } \{v_k^1, v_{k+1}^1\} \in E_d \text{ such that there is no } v_{k+1}^2 \in V_d \text{ for which } \{v_k^2, v_{k+1}^2\} \in E_d \text{ and } \pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2).$$

We say that a pair of separating walks of length k in G_d is *critical* if there does not exist a pair of separating walks of length k' in G_d for any $k' < k$.

Consider the graph G_5 in Figure 2. One example of a PSW in G_5 is the pair (\bar{v}_1, \bar{v}_2) , where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_7^i)$ for all $i = 1, 2$, and the sequence $\pi_5(v_j^i, v_{j-1}^i)$, $j = 1, 2, \dots, 7$, of generalised port numbers is 2, 2, 3, 3, 4, 4, 5. Observe that now node v_7^1 has a neighbour v_8^1 with $\pi_5(v_8^1, v_7^1) = 5$, but node v_7^2 does not have such a neighbour. The fact that the sequence grows slowly towards the parameter d is actually a general property of PSWs; this is one of the crucial ideas behind our proof.

The outline of the proof is as follows. First, we will prove auxiliary results concerning the graphs G_d and PSWs. These will enable us to obtain a lower bound for the length of PSWs. Then, we will show that this lower bound entails bisimilarity of the nodes $((1, 0))$ and $((2, 1))$ up to the respective distance. Since the overall proof is going to be a little hairy, we provide a chart of dependencies between the various lemmas in Figure 3. The first four lemmas follow quite easily from the definition of the graphs.

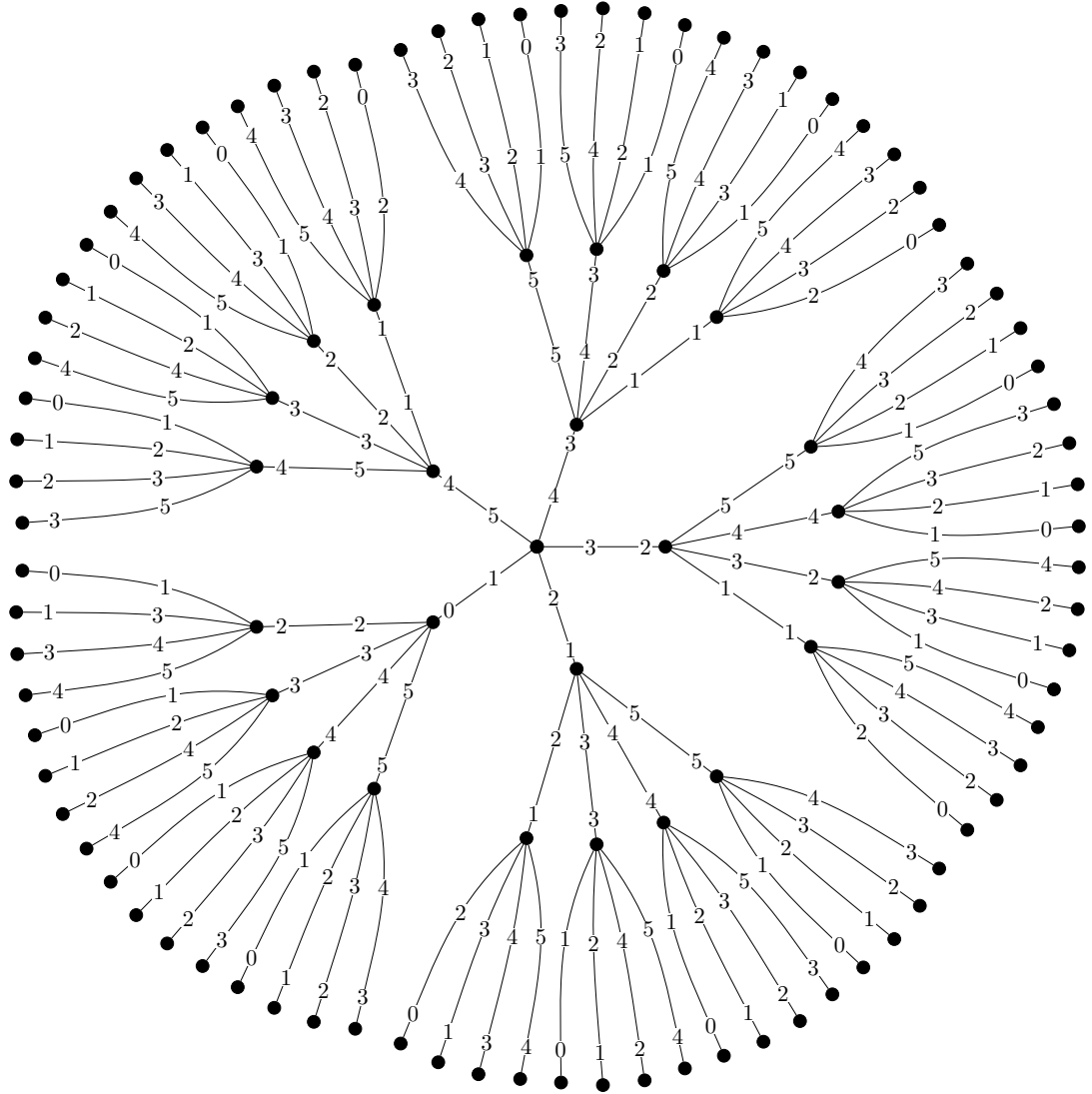


Figure 2: A part of the graph G_5 . The node in the centre is node \emptyset . The numbers pictured are outgoing port numbers.

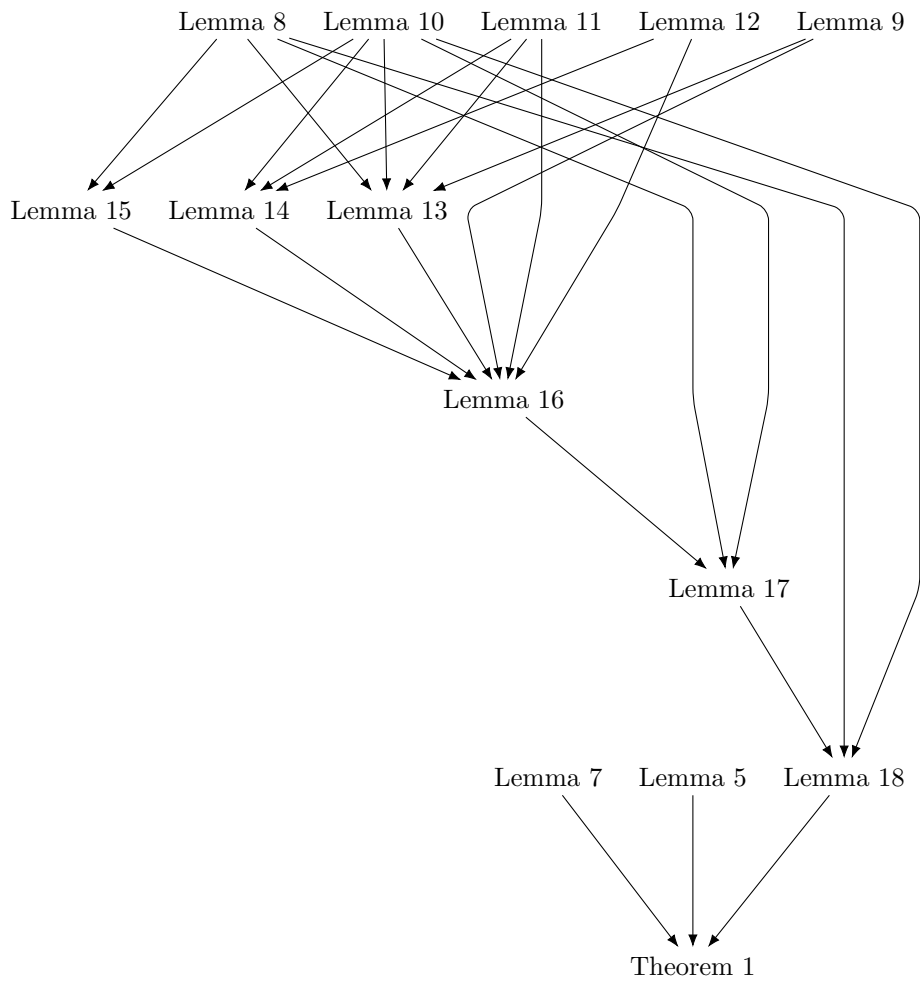


Figure 3: Dependencies between the lemmas that are needed in order to prove Theorem 1.

Lemma 8. *For each d , we have $\deg(v) \in \{1, d\}$ for all $v \in V_d$, and thus $G_d \in \mathcal{F}(d)$. Additionally, G_d is a subgraph of G_{d+1} .*

Proof. Consider a node $v = (a_1, a_2, \dots, a_i) \in V_d$. It follows from the definition that if $\{v, u\} \in E_d$, then u is either a parent or a child of v . If $i = 0$, v has no parents, and all its children are given by rule (G2). Hence $\deg(v) = d$. If $0 < i < 2d$, v has one parent, and all its children are given by rule (G3) or (G4). There are $d - 1$ children, and hence $\deg(v) = d$. If $i \geq 2d$, v has no children, and hence $\deg(v) = 1$.

It follows from the rules (G1)–(G4) that if $v \in V_d$, then also $v \in V_{d+1}$. Additionally, if $\{v, u\} \in E_d$, then clearly $\{v, u\} \in E_{d+1}$. This shows that G_d is a subgraph of G_{d+1} . \square

Lemma 9. *Let $v \in V_d$ and $a \in \{0, 1, \dots, d\}$. Then there is at most one node $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$.*

Proof. The claim follows immediately from rule (G2) and the way the numbers c_2^j are defined in rules (G3) and (G4). \square

A consequence of Lemma 9 is that in a walk, the successor of each node is uniquely determined by the port number from the successor to the node.

Lemma 10. *Let $v = (a_1, a_2, \dots, a_i) \in V_d$, where $i < 2d$. If v is odd, then for all $a \in \{1, 2, \dots, d\}$ there exists $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$. If v is even, then either for all $a \in \{0, 1, \dots, d - 1\}$ or for all $a \in \{0, 1, \dots, d - 2, d\}$ there exists $u \in V_d$ such that $\{v, u\} \in E_d$ and $\pi_d(u, v) = a$. In the case of even v and $a = d$, node u is the parent of node v .*

Proof. Observe that in rules (G3) and (G4), we always have $b_1 \in \{1, 2, \dots, d\}$. If v is odd, the claim follows from the way the numbers c_2^j are defined in rule (G3). If v is even, consider the application of rule (G4) to v . If $b_1 < d$, then c_2^j will range over all the elements in $\{0, 1, \dots, d - 1\} \setminus \{b_1\}$, and thus for all $a \in \{0, 1, \dots, d - 1\}$ there is a neighbour u such that $\pi_d(u, v) = a$. If $b_1 = d$, then c_2^j will range over all the elements in $\{0, 1, \dots, d - 2\}$, and thus for all $a \in \{0, 1, \dots, d - 2, d\}$ there is a neighbour u such that $\pi_d(u, v) = a$. We have always $c_2^j \neq d$, and hence the case $a = d$ is only possible if $b_1 = d$. It follows that if $\pi_d(u, v) = d$, u is the parent of v . \square

Lemma 10 implies that in a PSW, the last nodes of each walk must be even. Furthermore, one of the last nodes v must have a parent u with $\pi_d(u, v) = d$. It follows that we must have $v \in V_d \setminus V_{d-1}$.

Lemma 11. *Let $\{v, u\} \in E_{d+1} \setminus E_d$ be such that $v \in V_d$. Then u is a child of v . If v is odd, then $\pi_{d+1}(v, u) = \pi_{d+1}(u, v) = d + 1$. If v is even, then $\pi_{d+1}(v, u) = d + 1$ and $\pi_{d+1}(u, v) \in \{d - 1, d\}$.*

Proof. Since $\{v, u\} \in E_{d+1}$, u is either the parent or a child of v . If it was the parent, we would have $u \in V_d$ and thus $\{v, u\} \in E_d$, a contradiction. Hence $u \in V_{d+1} \setminus V_d$ is a child of v . If $v = (a_1, a_2, \dots, a_i)$ is odd, u is given by rule (G3) in the definition of G_{d+1} . Since $(a_1, a_2, \dots, a_{i+1}^j) \in V_d$ for all $j = 1, 2, \dots, d - 1$, we have $u = (a_1, a_2, \dots, a_{i+1}^d)$. As $v \in V_d$, we have $b_1, b_2^+ \leq d$, and thus $c_1^d = c_2^d = d + 1$. This implies $\pi_{d+1}(v, u) = \pi_{d+1}(u, v) = d + 1$. If $v = (a_1, a_2, \dots, a_i)$ is even, u is given by rule (G4) in the definition of G_{d+1} . Again, we have $u = (a_1, a_2, \dots, a_{i+1}^d)$, $b_1, b_2 \leq d$ and thus $c_1^d = d + 1$. If $b_1 = d$, then $c_2^d = d - 1$, otherwise $c_2^d = d$. This implies $\pi_{d+1}(v, u) = d + 1$ and $\pi_{d+1}(u, v) \in \{d - 1, d\}$. \square

With the above observations out of the way, we now go forward with more powerful results.

Lemma 12. *Let (\bar{v}_1, \bar{v}_2) , where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$, be a PSW in G_d . If for some $\ell \in \{0, 1, \dots, k - 1\}$ the node $v_{\ell+1}^i$ is a child of node v_ℓ^i for all $i = 1, 2$, and we have $\pi_d(v_\ell^1, v_{\ell+1}^1) = \pi_d(v_\ell^2, v_{\ell+1}^2)$, then (\bar{v}_1, \bar{v}_2) is not a critical PSW in G_d .*

Proof. Suppose that for all $m = \ell + 2, \ell + 3, \dots, k$ we have $v_m^1 \neq v_\ell^1$ or $v_m^2 \neq v_\ell^2$. By assumption, $v_{\ell+1}^1$ and $v_{\ell+1}^2$ are of the same type. Consider the definition of G_d . Now it is easy to show by induction on m that nodes v_m^1 and v_m^2 are of the same type for all $m = \ell + 1, \ell + 2, \dots, k$. Since $k \leq 2d - 3$, both v_k^1 and v_k^2 have child nodes. It follows that if v_{k+1}^1 is a neighbour of v_k^1 , there is a neighbour v_{k+1}^2 of v_k^2 such that $\pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2)$. Thus (\bar{v}_1, \bar{v}_2) is not a PSW in G_d , a contradiction.

Now $v_m^1 = v_\ell^1$ and $v_m^2 = v_\ell^2$ for some $m \in \{\ell + 2, \ell + 3, \dots, k\}$. Let

$$\bar{v}'_i = (v_0^i, v_1^i, \dots, v_\ell^i, v_{m+1}^i, v_{m+2}^i, \dots, v_k^i)$$

for all $i = 1, 2$. Then (\bar{v}'_1, \bar{v}'_2) is a PSW of length $k - m + \ell \leq k - (\ell + 2) + \ell = k - 2 < k$ in G_d and hence (\bar{v}_1, \bar{v}_2) is not critical. \square

Lemma 13. *Let (\bar{v}_1, \bar{v}_2) be a PSW of length k in G_d . Then there is a PSW of length $k + 2$ in G_{d+1} .*

Proof. Let $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$. By definition, there is a neighbour $u \in V_d$ of v_k^1 such that for each neighbour $w \in V_d$ of v_k^2 we have $\pi_d(u, v_k^1) \neq \pi_d(w, v_k^2)$. Lemma 10 implies that v_k^1 and v_k^2 are even, $\pi_d(u, v_k^1) \in \{d - 1, d\}$, and there is a neighbour $w \in V_d$ of v_k^2 for which $\pi_d(w, v_k^2) \in \{d - 1, d\} \setminus \{\pi_d(u, v_k^1)\}$. That is, we have $\pi_d(u, v_k^1) = d$ or $\pi_d(w, v_k^2) = d$. Without loss of generality, we can assume $\pi_d(u, v_k^1) = d$ and thus $\pi_d(w, v_k^2) = d - 1$.

Lemma 8 implies that $\deg_{G_d}(u) = \deg_{G_d}(v_k^2) = d$ and $\deg_{G_{d+1}}(u) = \deg_{G_{d+1}}(v_k^2) = d + 1$. Hence there are nodes $x, y \in V_{d+1} \setminus V_d$ such that $\{u, x\} \in E_{d+1} \setminus E_d$ and $\{v_k^2, y\} \in E_{d+1} \setminus E_d$. Note that $u, v_k^2 \in V_d$, u is odd and v_k^2 is even. It follows from Lemma 11 that $\pi_{d+1}(u, x) = \pi_{d+1}(x, u) = d + 1$, $\pi_{d+1}(v_k^2, y) = d + 1$ and $\pi_{d+1}(y, v_k^2) \in \{d - 1, d\}$. Since $\pi_{d+1}(w, v_k^2) = \pi_d(w, v_k^2) = d - 1$ and $w \neq y$, Lemma 9 implies that $\pi_{d+1}(y, v_k^2) = d$.

Now we can extend the walks \bar{v}_1 and \bar{v}_2 . Set $\bar{v}'_1 = (v_0^1, v_1^1, \dots, v_k^1, u, x)$ and $\bar{v}'_2 = (v_0^2, v_1^2, \dots, v_k^2, y, v_k^2)$. We have $\pi_{d+1}(u, v_k^1) = d = \pi_{d+1}(y, v_k^2)$ and $\pi_{d+1}(x, u) = d + 1 = \pi_{d+1}(v_k^2, y)$, as required. Furthermore, node x has neighbour u for which $\pi_{d+1}(u, x) = d + 1$. Suppose that there is a neighbour u' of v_k^2 for which $\pi_{d+1}(u', v_k^2) = d + 1$. Now Lemma 10 implies that u' is the parent of v_k^2 . But since $v_k^2 \in V_d$, we have also $u' \in V_d$, and hence $\pi_{d+1}(u', v_k^2) \leq d$, a contradiction. Similarly, node v_k^2 has neighbour y for which $\pi_{d+1}(y, v_k^2) = d$, but $\pi_{d+1}(u, x) = d + 1$ together with Lemma 10 implies that there is no neighbour y' of x for which $\pi_{d+1}(y', x) = d$. This shows that (\bar{v}'_1, \bar{v}'_2) is a PSW of length $k + 2$ in G_{d+1} . \square

Lemma 14. *Let (\bar{v}_1, \bar{v}_2) , where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$, be a critical PSW in G_d . Then we have $v_{k-1}^i \in V_d \setminus V_{d-1}$ for some $i \in \{1, 2\}$.*

Proof. Lemma 10 implies that v_k^1 and v_k^2 are even, and for some $i \in \{1, 2\}$ node v_k^i has a parent u such that $\pi_d(u, v_k^i) = d$. If $v_k^i \in V_{d-1}$, then also $u \in V_{d-1}$ and hence $\pi_d(u, v_k^i) \leq d - 1$, a contradiction. Therefore $v_k^i \in V_d \setminus V_{d-1}$.

Suppose that $v_{k-1}^j \in V_{d-1}$ for all $j = 1, 2$. Since $v_k^i \in V_d \setminus V_{d-1}$, we have $\{v_{k-1}^i, v_k^i\} \in E_d \setminus E_{d-1}$. Lemma 11 implies that v_k^i is a child of v_{k-1}^i and $\pi_d(v_{k-1}^i, v_k^i) = \pi_d(v_k^i, v_{k-1}^i) = d$. Let $j \in \{1, 2\} \setminus \{i\}$. As $\pi_d(v_k^j, v_{k-1}^j) = \pi_d(v_k^i, v_{k-1}^i) = d$, we have $\{v_{k-1}^j, v_k^j\} \in E_d \setminus E_{d-1}$ and thus v_k^j is a child of v_{k-1}^j and $\pi_d(v_{k-1}^j, v_k^j) = \pi_d(v_k^j, v_{k-1}^j) = d$. Now it follows from Lemma 12 that (\bar{v}_1, \bar{v}_2) is not a critical PSW in G_d , a contradiction. \square

Lemma 15. *Let (\bar{v}_1, \bar{v}_2) , where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$, be a PCW in G_d . If (\bar{v}_1, \bar{v}_2) is not a PSW in G_d , then for each neighbour $v_{k+1}^1 \in V_d$ of v_k^1 there is a neighbour $v_{k+1}^2 \in V_d$ of v_k^2 such that $\pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2)$, and vice versa.*

Proof. Since (\bar{v}_1, \bar{v}_2) is not a PSW, condition (W3) does not hold. This is equivalent to the first claim. For the second claim, assume that v_{k+1}^2 is a neighbour of v_k^2 . Suppose that there is no neighbour v_{k+1}^1 of v_k^1 such that $\pi_d(v_{k+1}^1, v_k^1) = \pi_d(v_{k+1}^2, v_k^2)$. Now it follows from Lemma 8 and Lemma 10 that v_k^1 and v_k^2 are even and $\pi_d(v_{k+1}^2, v_k^2) \in \{d-1, d\}$. We also obtain from Lemma 10 that there is a neighbour u of v_k^1 for which $\pi_d(u, v_k^1) \in \{d-1, d\} \setminus \{\pi_d(v_{k+1}^2, v_k^2)\}$. Now u is a neighbour of v_k^1 such that there is no neighbour w of v_k^2 for which $\pi_d(u, v_k^1) = \pi_d(w, v_k^2)$, a contradiction. \square

Now we are ready to prove the following lemma, which is the main ingredient of the proof of Theorem 1. The underlying idea is that the generalised port numbers along the walks have to grow slowly. Put otherwise, each prefix of a critical PSW must be contained in a subgraph G_d for a sufficiently small value of d .

Lemma 16. *Let (\bar{v}_1, \bar{v}_2) , where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$, be a critical PSW in G_d . Then (\bar{v}'_1, \bar{v}'_2) , where $\bar{v}'_i = (v_0^i, v_1^i, \dots, v_{k-2}^i)$ for all $i = 1, 2$, is a PSW in G_{d-1} .*

Proof. First, suppose that $\{v_\ell^i, v_{\ell+1}^i\} \in E_{d-1}$ for all $i = 1, 2$ and $\ell = 0, 1, \dots, k-3$ but that (\bar{v}'_1, \bar{v}'_2) is not a PSW in G_{d-1} . Assume that $\{v_{k-2}^i, v_{k-1}^i\} \in E_{d-1}$ for some $i \in \{1, 2\}$ and let $j \in \{1, 2\} \setminus \{i\}$. It follows from Lemma 15 that there is a neighbour $u \in V_{d-1}$ of v_{k-2}^j such that $\pi_{d-1}(u, v_{k-2}^j) = \pi_{d-1}(v_{k-1}^i, v_{k-2}^i)$. Now Lemma 9 implies that $u = v_{k-1}^j$ and hence we have $v_{k-1}^i, v_{k-1}^j \in V_{d-1}$. Then we can use Lemma 14 to obtain that (\bar{v}_1, \bar{v}_2) is not a critical PSW in G_d , a contradiction.

Let us then assume that $\{v_{k-2}^i, v_{k-1}^i\} \in E_d \setminus E_{d-1}$ for all $i = 1, 2$. As $v_{k-2}^i \in V_{d-1}$ for all $i = 1, 2$, Lemma 11 implies that v_{k-1}^i is a child of v_{k-2}^i and $\pi_d(v_{k-2}^i, v_{k-1}^i) = d = \pi_d(v_{k-2}^j, v_{k-1}^j)$ for all $i = 1, 2$. But now we can apply Lemma 12 to see that (\bar{v}_1, \bar{v}_2) is not a critical PSW in G_d , a contradiction. We have now shown that if $\{v_\ell^i, v_{\ell+1}^i\} \in E_{d-1}$ for all $i = 1, 2$ and $\ell = 0, 1, \dots, k-3$, then (\bar{v}'_1, \bar{v}'_2) is a PSW in G_{d-1} .

Then, suppose that $\{v_\ell^i, v_{\ell+1}^i\} \in E_d \setminus E_{d-1}$ for some $i \in \{1, 2\}$ and $\ell \in \{0, 1, \dots, k-3\}$. Let m be the smallest value of ℓ for which this holds. Let $j \in \{1, 2\} \setminus \{i\}$. If m is even, then the node $v_m^i \in V_{d-1}$ is odd, and by Lemma 11 we have that $\pi_d(v_m^i, v_{m+1}^i) = \pi_d(v_{m+1}^i, v_m^i) = d$ and that v_{m+1}^i is a child of v_m^i . Since $\pi_d(v_{m+1}^j, v_m^j) = \pi_d(v_{m+1}^i, v_m^i) = d$, we obtain $\{v_m^j, v_{m+1}^j\} \in E_d \setminus E_{d-1}$. As $v_m^j \in V_{d-1}$ is odd, Lemma 11 yields that $\pi_d(v_m^j, v_{m+1}^j) = \pi_d(v_{m+1}^j, v_m^j) = d$ and that v_{m+1}^j is a child of v_m^j . Lemma 12 then implies that (\bar{v}_1, \bar{v}_2) is not a critical PSW in G_d , a contradiction.

To complete the proof, assume that m is odd. Recall that $\{v_m^i, v_{m+1}^i\} \in E_d \setminus E_{d-1}$. If also $\{v_m^j, v_{m+1}^j\} \in E_d \setminus E_{d-1}$, we can again use Lemma 11 to get that v_{m+1}^i and v_{m+1}^j are children of v_m^i and v_m^j , respectively, and that $\pi_d(v_m^i, v_{m+1}^i) = d = \pi_d(v_m^j, v_{m+1}^j)$. Now Lemma 12 yields a contradiction. If $\{v_m^j, v_{m+1}^j\} \in E_{d-1}$, let $\bar{v}''_\ell = (v_\ell^j, v_{\ell+1}^j, \dots, v_m^j)$ for all $\ell = 1, 2$. The pair $(\bar{v}'_1, \bar{v}''_2)$ is a PSW in G_{d-1} , because otherwise by using a similar argument as above we would obtain that $\{v_m^i, v_{m+1}^i\} \in E_{d-1}$, a contradiction. But now we can use Lemma 13 to get a PSW of length $m+2 \leq (k-3)+2 = k-1$ in G_d , which contradicts the criticality of (\bar{v}_1, \bar{v}_2) . \square

Having proved Lemma 16, the following result now follows by induction.

Lemma 17. *Let (\bar{v}_1, \bar{v}_2) be a PSW of length k in G_d . Then $k \geq 2d-3$.*

Proof. We use induction on d . Let $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$. It follows from Lemma 8 and Lemma 10 that v_k^i is even for all $i = 1, 2$, and thus k is odd. Hence we have $k \geq 1$. If $d = 2$, we have shown that $k \geq 2d-3$.

For the inductive step, suppose that the claim holds for $d = q$ and that (\bar{v}_1, \bar{v}_2) is a PSW of length k in G_{q+1} . Now there is a critical PSW (\bar{u}_1, \bar{u}_2) of length $\ell \leq k$ in G_{q+1} , where $\bar{u}_i = (u_0^i, u_1^i, \dots, u_\ell^i)$ for all $i = 1, 2$. Lemma 16 implies that (\bar{u}'_1, \bar{u}'_2) , where $\bar{u}'_i = (u_0^i, u_1^i, \dots, u_{\ell-2}^i)$ for all $i = 1, 2$, is a PSW of length $\ell-2$ in G_q . By the inductive hypothesis we obtain $\ell-2 \geq 2q-3$.

It follows that $k \geq \ell \geq 2q - 1 = 2(q + 1) - 3$. Hence we have shown that the claim holds for $d = q + 1$. \square

Now we just need to show that the lower bound for the length of PSWs implies bisimilarity up to the respective distance, and we are mostly done.

Lemma 18. *We have $((1, 0)) \leftrightarrow_{2d-3}^{\mathcal{SV}} ((2, 1))$, that is, the nodes $((1, 0))$ and $((2, 1))$ of G_d are $(2d - 3)$ - \mathcal{SV} -bisimilar.*

Proof. If we have $((1, 0)) \leftrightarrow_k^{\mathcal{SV}} ((2, 1))$ for arbitrarily large k , the claim is clearly true. Otherwise, let k be the largest integer for which we have $((1, 0)) \leftrightarrow_k^{\mathcal{SV}} ((2, 1))$. We will show that $k \geq 2d - 3$.

Let $v_0^1 = ((1, 0))$ and $v_0^2 = ((2, 1))$. Suppose then that $\ell \in \{0, 1, \dots, k - 1\}$ and that v_ℓ^1 and v_ℓ^2 have been defined. Furthermore, suppose that $k - \ell$ is the largest integer m for which $v_\ell^1 \leftrightarrow_m^{\mathcal{SV}} v_\ell^2$ holds. If for each neighbour u of v_ℓ^1 there was a neighbour w of v_ℓ^2 , and vice versa, such that $u \leftrightarrow_{k-\ell}^{\mathcal{SV}} w$ and $\pi_d(u, v_\ell^1) = \pi_d(w, v_\ell^2)$, then by Definition 4 we would have $v_\ell^1 \leftrightarrow_{k-\ell+1}^{\mathcal{SV}} v_\ell^2$, a contradiction. Thus for some $i \in \{1, 2\}$ and $j \in \{1, 2\} \setminus \{i\}$ there is a neighbour u of v_ℓ^i such that there is no neighbour w of v_ℓ^j for which the given condition holds. However, since $v_\ell^i \leftrightarrow_{k-\ell}^{\mathcal{SV}} v_\ell^j$, we can choose neighbour w so that $u \leftrightarrow_{k-\ell-1}^{\mathcal{SV}} w$ and $\pi_d(u, v_\ell^i) = \pi_d(w, v_\ell^j)$. Now we can define $v_{\ell+1}^i = u$ and $v_{\ell+1}^j = w$. We have shown that $k - \ell - 1 = k - (\ell + 1)$ is the largest integer m for which $v_{\ell+1}^i \leftrightarrow_m^{\mathcal{SV}} v_{\ell+1}^j$ holds.

The above recursive definition yields a pair (\bar{v}_1, \bar{v}_2) of walks, where $\bar{v}_i = (v_0^i, v_1^i, \dots, v_k^i)$ for all $i = 1, 2$. Clearly conditions (W1) and (W2) hold. Additionally, we know that $k - k = 0$ is the largest integer m for which we have $v_k^1 \leftrightarrow_m^{\mathcal{SV}} v_k^2$. However, if $k \leq 2d - 3$, then for each neighbour u of v_k^1 and w of v_k^2 we have $\deg(u) = \deg(w)$ and hence $u \leftrightarrow_0^{\mathcal{SV}} w$. It follows that for some $i \in \{1, 2\}$ and $j \in \{1, 2\} \setminus \{i\}$ there is a neighbour u of v_k^i such that there is no neighbour w of v_k^j for which $\pi_d(u, v_k^i) = \pi_d(w, v_k^j)$. If $i = 1$ and $j = 2$, this is equivalent to condition (W3). Otherwise, we use Lemma 8 and Lemma 10 to swap the roles of i and j in a similar manner as in the proof of Lemma 15.

In conclusion, we have shown that if $k \leq 2d - 3$, then (\bar{v}_1, \bar{v}_2) is a PSW of length k in G_d . Now Lemma 17 implies that $k = 2d - 3$. If $k > 2d - 3$, the claim is trivially true. \square

Remark 19. Lemma 18 can also be viewed from a game-theoretic perspective. When considering a game played by *Spoiler* and *Duplicator* starting from the nodes $((1, 0))$ and $((2, 1))$, the pair of sequences consisting of the nodes chosen by the players is a PSW. Then, the lower bound on the length of PSWs implies that *Duplicator* has a *winning strategy* in the $(2d - 3)$ -round bisimulation game. For more details on bisimulation games, see Blackburn, Bentham and Wolter [2].

To prove Theorem 1, we want the root node \emptyset to receive the same messages from its neighbours $((1, 0))$ and $((2, 1))$. Lemma 18 shows that they are $(2d - 3)$ - \mathcal{SV} -bisimilar, but this is not enough: they also need to have identical outgoing port numbers towards node \emptyset . We will now define a port numbering of G_d based on the generalised port numbering p_d . Let $f: \{0, 1, \dots, d\} \rightarrow [d]$ be a function such that $f(0) = 1$ and $f(i) = i$ for $i \geq 1$. If $p_d(v, i) = (u, j)$ for some nodes v, u and port numbers i, j , we define $p'_d(v, (f(i))) = (u, f(j))$. Due to the fact that in rule (G3) of the definition of G_d we used b_2^+ instead of b_2 , no node has both 0 and 1 as port numbers in p_d . It follows that p'_d is a bijection from the set of input ports to the set of output ports, and the set of outgoing as well as incoming port numbers for each node v is $\{1, 2, \dots, \deg(v)\}$. Observe that $p'_d(((1, 0)), 1) = (\emptyset, 1)$ and $p'_d(((2, 1)), 1) = (\emptyset, 2)$. Now we can apply Lemma 7 to see that the $(2d - 3)$ - \mathcal{SV} -bisimilarity still holds, that is, we have $(G_d, ((1, 0)), p'_d) \leftrightarrow_{2d-3}^{\mathcal{SV}} (G_d, ((2, 1)), p'_d)$.

Let $\mathcal{A} \in \mathcal{SV}$ be an arbitrary algorithm and $\Delta \geq 2$. Let $G = G_\Delta$, $p = p'_\Delta$, $v = \emptyset$, $u = ((1, 0))$ and $w = ((2, 1))$. Consider the execution of \mathcal{A} in (G, p) . Lemma 5 implies that the state of \mathcal{A} in the nodes u and w is identical in each round $r = 0, 1, \dots, 2\Delta - 3$. Furthermore, we have $\pi(u, v) = 1 = \pi(w, v)$. It follows that u and w send the same message to node v in each round $r + 1 = 1, 2, \dots, 2\Delta - 2$. This concludes the proof of Theorem 1.

Remark 20. We could as well show that the nodes $((1, 0))$ and $((2, 1))$ are $(2d - 3)$ -bisimilar with respect to the class \mathcal{MV} of algorithms, with only minor changes to the proof of Lemma 18. However, this would not make any difference in the end, since we need to consider an algorithm in \mathcal{SV} for the root node to lose the multiplicities of messages it receives from its neighbours.

4 Separation by a Graph Problem

Theorem 1 shows that the simulation algorithm is optimal in a certain sense. However, since we are interested in graph problems, we want to separate the classes \mathcal{SV} and \mathcal{MV} by one. The following theorem states that we can do this, and the lower bound is still linear in Δ .

Theorem 2. *There is a graph problem Π that can be solved in one round by an algorithm in \mathcal{MV} but that requires at least time T , where $T(n, \Delta) \geq \Delta$ for all $\Delta \geq 2$, when solved by an algorithm in \mathcal{SV} .*

Let us first define formally the graph problem Π . We will be working with graphs where each node is given as a local input one of three colours: black (B), white (W) or grey (G). For each graph (G, f) with local input from the set $\{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$, the set $\Pi(G, f)$ of solutions consists of mappings $S: V \rightarrow \{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$ such that for each $v \in V$, $S(v)$ is one of the local inputs having the highest multiplicity among the neighbours of v . For example, if node v has four neighbours of colour B, four neighbours of colour W and two neighbours of colour G, then for each solution S we have $S(v) = \mathbf{B}$ or $S(v) = \mathbf{W}$.

There is an algorithm in \mathcal{MV} —and, in fact, in \mathcal{MB} —that solves problem Π in only one communication round: Each node broadcasts its own colour to all its neighbours. Then, each node counts the multiplicity of each message it received and outputs the one with the highest multiplicity. Showing that this cannot be solved by any algorithm in \mathcal{SV} in less than Δ communication rounds will require somewhat more work. Luckily, we can handle the most tricky part of the proof by making use of the proof of Theorem 1 in a black-box manner.

We start by defining for each $d = 2, 3, \dots$ two graphs, $H_{\mathbf{B},d} = (V_{\mathbf{B},d}, E_{\mathbf{B},d})$ and $H_{\mathbf{W},d} = (V_{\mathbf{W},d}, E_{\mathbf{W},d})$. The constructions can be seen as extensions of the graph G_d defined earlier, but now each node is coloured with one of the three colours: black (B), white (W) or grey (G). Colours B and W can be thought of as complements of each other; we write $\bar{\mathbf{B}} = \mathbf{W}$ and $\bar{\mathbf{W}} = \mathbf{B}$. Again, we define $V_{\mathbf{B},d}$ recursively:

$$(H1) \quad \emptyset \in V_{\mathbf{B},d}.$$

$$(H2) \quad ((1, 0, \mathbf{B})), ((2, 1, \mathbf{B})), ((3, 2, \mathbf{B})), ((4, 3, \mathbf{B})), \dots, ((d, d-1, \mathbf{B})) \in V_{\mathbf{B},d}.$$

$$(H3) \quad ((2, 1, \mathbf{W})), ((3, 2, \mathbf{W})), ((4, 3, \mathbf{W})), \dots, ((d, d-1, \mathbf{W})) \in V_{\mathbf{B},d}.$$

$$(H4) \quad \text{If } (a_1, a_2, \dots, a_i) \in V_{\mathbf{B},d}, \text{ where } i \text{ is odd and } i < 2d, \text{ then } (a_1, a_2, \dots, a_{i+1}^j) \in V_{\mathbf{B},d} \text{ for all } j = 1, 2, \dots, d-1, \text{ where } a_{i+1}^j = (c_1^j, c_2^j, \mathbf{G}) \text{ is defined as follows. Let } (b_1, b_2, C) = a_i, \text{ where } C \in \{\mathbf{B}, \mathbf{W}\}, \text{ and } b_2^+ = 1 \text{ if } b_2 = 0, b_2^+ = b_2 \text{ otherwise. Define}$$

$$c_1^j = \min(\{1, 2, \dots, d\} \setminus \{b_2^+, c_1^1, c_1^2, \dots, c_1^{j-1}\}),$$

$$c_2^j = \min(\{1, 2, \dots, d\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}).$$

$$(H5) \quad \text{If } (a_1, a_2, \dots, a_i) \in V_{\mathbf{B},d}, \text{ where } i \text{ is even and } i < 2d, \text{ then } (a_1, a_2, \dots, a_{i+1}^j) \in V_{\mathbf{B},d} \text{ for all } j = 1, 2, \dots, d-1, \text{ where } a_{i+1}^j = (c_1^j, c_2^j, C) \text{ is defined as follows. Let } (d_1, d_2, C) = a_{i-1}, \text{ where } C \in \{\mathbf{B}, \mathbf{W}\}, \text{ and } (b_1, b_2, \mathbf{G}) = a_i. \text{ Define}$$

$$c_1^j = \min(\{1, 2, \dots, d\} \setminus \{b_2, c_1^1, c_1^2, \dots, c_1^{j-1}\}),$$

$$c_2^j = \min(\{0, 1, \dots, d-1\} \setminus \{b_1, c_2^1, c_2^2, \dots, c_2^{j-1}\}).$$

(H6) If $(a_1, a_2, \dots, a_i) \in V_{\mathbf{B},d}$, where i is even and $i < 2d$, then $(a_1, a_2, \dots, a_{i+1}^j) \in V_{\mathbf{B},d}$ for all $j = 1, 2, \dots, d-1$, where $a_{i+1}^j = (c_1^j, c_2^j, \overline{C})$ is defined as follows. Let $(d_1, d_2, C) = a_{i-1}$, where $C \in \{\mathbf{B}, \mathbf{W}\}$. Define

$$\begin{aligned} c_1^j &= \min(\{2, 3, \dots, d\} \setminus \{c_1^1, c_1^2, \dots, c_1^{j-1}\}), \\ c_2^j &= \min(\{1, 2, \dots, d-1\} \setminus \{c_2^1, c_2^2, \dots, c_2^{j-1}\}). \end{aligned}$$

The set $E_{\mathbf{B},d}$ of edges consists of all pairs $\{v, u\}$, where $v = (a_1, a_2, \dots, a_i) \in V_{\mathbf{B},d}$ and $u = (a_1, a_2, \dots, a_i, a_{i+1}) \in V_{\mathbf{B},d}$ for some $i \in \{0, 1, \dots\}$. The sets $V_{\mathbf{W},d}$ and $E_{\mathbf{W},d}$ are given by the same definition by replacing every occurrence of \mathbf{B} with \mathbf{W} and vice versa. See Figures 4 and 5 for illustrations. By rearranging the branches of the trees, we observe that actually the only difference between $H_{\mathbf{B},d}$ and $H_{\mathbf{W},d}$ is the colours in the branch that starts with the node $((1, 0, C))$.

In this proof we work with the graphs $H_{\mathbf{B},d}$ and $H_{\mathbf{W},d}$ for a fixed value of d . Hence, to simplify notation, we will write $H_{\mathbf{B}}$ and $H_{\mathbf{W}}$ from now on.

We define colourings $f_{\mathbf{B}}: V_{\mathbf{B}} \rightarrow \{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$ and $f_{\mathbf{W}}: V_{\mathbf{W}} \rightarrow \{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$ as follows. If $v = (a_1, a_2, \dots, a_i) \in V_C$ for some $C \in \{\mathbf{B}, \mathbf{W}\}$ and $i \geq 1$, and we have $a_i = (b_1, b_2, C')$, set $f_C(v) = C'$. If $v = \emptyset \in V_C$, set $f_C(v) = \mathbf{G}$. Notice that for each solution $S \in \Pi(H_{\mathbf{B}}, f_{\mathbf{B}})$ we have $S(\emptyset) = \mathbf{B}$ and for each solution $S \in \Pi(H_{\mathbf{W}}, f_{\mathbf{W}})$ we have $S(\emptyset) = \mathbf{W}$.

Our port numbers are pairs (a, C) , where $a \in \{0, 1, \dots, d\}$ and $C \in \{\mathbf{B}, \mathbf{W}, \mathbf{G}\}$. Generalised port numberings $p_{\mathbf{B}}$ and $p_{\mathbf{W}}$ for $H_{\mathbf{B}}$ and $H_{\mathbf{W}}$, respectively, are defined as follows. Let $v = (a_1, a_2, \dots, a_i)$ and $u = (a_1, a_2, \dots, a_{i+1})$, where $a_{i+1} = (b_1, b_2, C)$, be nodes. Note that $f_{\mathbf{B}}(u) = f_{\mathbf{W}}(u) = C$. If $C \in \{\mathbf{B}, \mathbf{W}\}$, define

$$\begin{aligned} p_{\mathbf{B}}(v, (b_1, C)) &= p_{\mathbf{W}}(v, (b_1, C)) = (u, (b_2, \mathbf{G})), \\ p_{\mathbf{B}}(u, (b_2, \mathbf{G})) &= p_{\mathbf{W}}(u, (b_2, \mathbf{G})) = (v, (b_1, C)). \end{aligned}$$

If $C = \mathbf{G}$, let $C' = f_{\mathbf{B}}(v) = f_{\mathbf{W}}(v)$ and define

$$\begin{aligned} p_{\mathbf{B}}(v, (b_1, \mathbf{G})) &= p_{\mathbf{W}}(v, (b_1, \mathbf{G})) = (u, (b_2, C')), \\ p_{\mathbf{B}}(u, (b_2, C')) &= p_{\mathbf{W}}(u, (b_2, C')) = (v, (b_1, \mathbf{G})). \end{aligned}$$

Next we will define induced subgraphs $\hat{H}_{\mathbf{B}}$ and $\hat{H}_{\mathbf{W}}$ of $H_{\mathbf{B}}$ and $H_{\mathbf{W}}$, respectively. For $C \in \{\mathbf{B}, \mathbf{W}\}$, the vertex set \hat{V}_C of \hat{H}_C consists of all vertices $(a_1, a_2, \dots, a_i) \in V_C$ such that $f_C((a_1, a_2, \dots, a_j)) \in \{C, \mathbf{G}\}$ for all $j \in \{0, 1, \dots, i\}$. That is, a node v of H_C is in the subgraph \hat{H}_C if and only if each node in the unique path from the root node \emptyset to node v is either grey or of colour C . For each $v = (a_1, a_2, \dots, a_i) \in V_C$ we denote the corresponding node of \hat{H}_C by $\hat{v} = (a_1, a_2, \dots, a_i) \in \hat{V}_C$.

For each $C \in \{\mathbf{B}, \mathbf{W}\}$, define a mapping $g_C: \hat{V}_C \rightarrow V_d$ as follows. Assume $\hat{v} = (a_1, a_2, \dots, a_i) \in \hat{V}_C$, where $a_j = (b_1^j, b_2^j, C_j)$ for each j . Now set $g_C(\hat{v}) = (a'_1, a'_2, \dots, a'_i)$, where $a'_j = (b_1^j, b_2^j)$ for each j . By observing that the subgraph \hat{H}_C is given by the rules (H1), (H2), (H4) and (H5) in the definition of H_C , and how they correspond to the rules (G1)–(G4) in the definition of G_d , one can see that g_C is a bijection, and in fact an isomorphism, between \hat{H}_C and G_d . We can use g_C to move bisimilarity results from G_d to \hat{H}_C , as the following lemma shows.

Lemma 21. *Let $C \in \{\mathbf{B}, \mathbf{W}\}$, $r \in \mathbb{N}$ and $\hat{v}, \hat{u} \in \hat{V}_C$. If $g_C(\hat{v}) \xleftrightarrow[r]{S^V} g_C(\hat{u})$ and $f_C(\hat{v}) = f_C(\hat{u})$, then $\hat{v} \xleftrightarrow[r]{S^V} \hat{u}$.*

Proof. The proof is by induction on r . Given the inductive hypothesis and conditions (B1)–(B3) of Definition 4 for $g_C(\hat{v})$ and $g_C(\hat{u})$, it is quite straightforward to check that the conditions also hold for \hat{v} and \hat{u} . \square

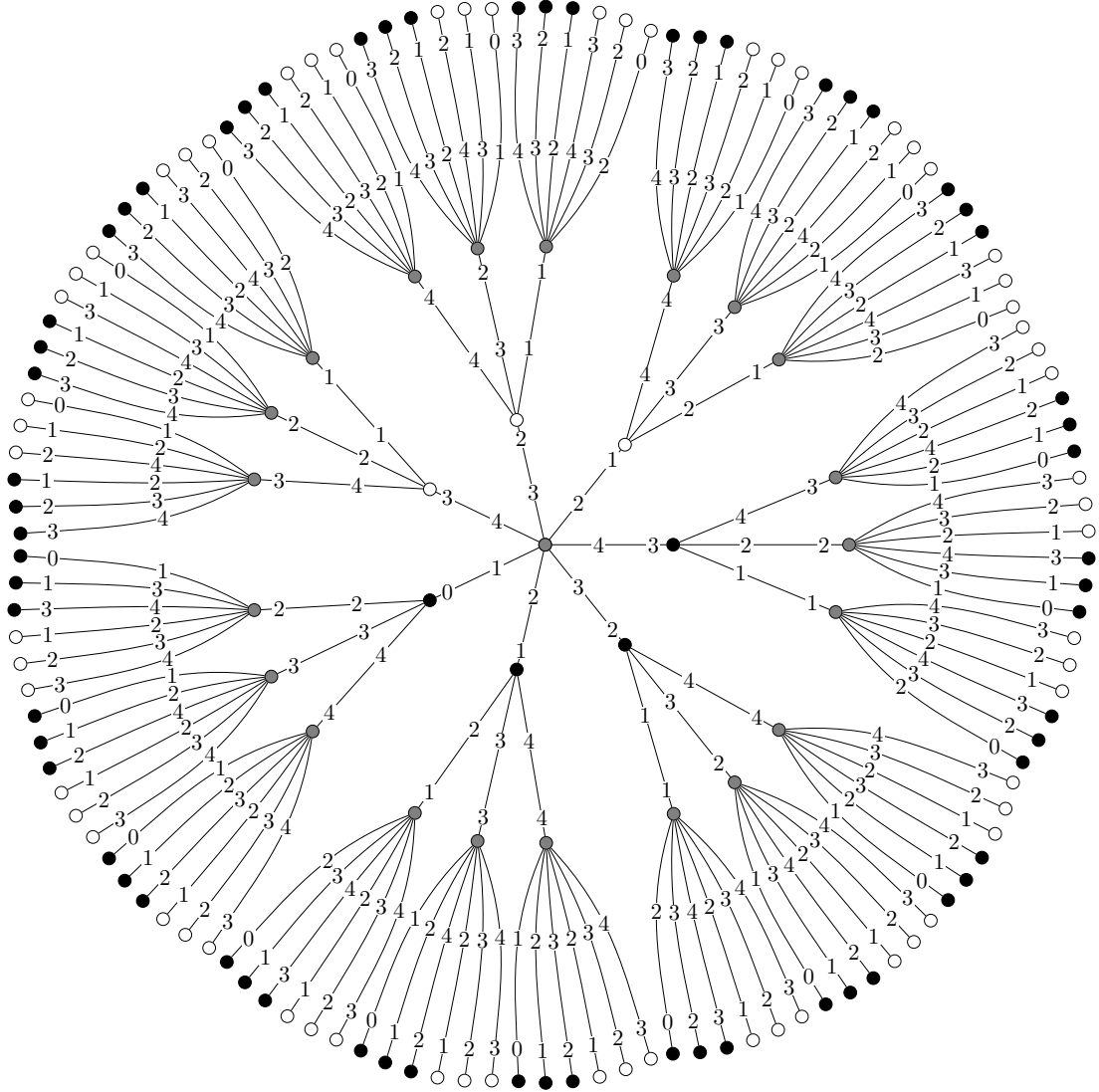


Figure 4: A part of the graph $H_{B,4}$. The node in the centre is node \emptyset . The numbers pictured are outgoing port numbers.

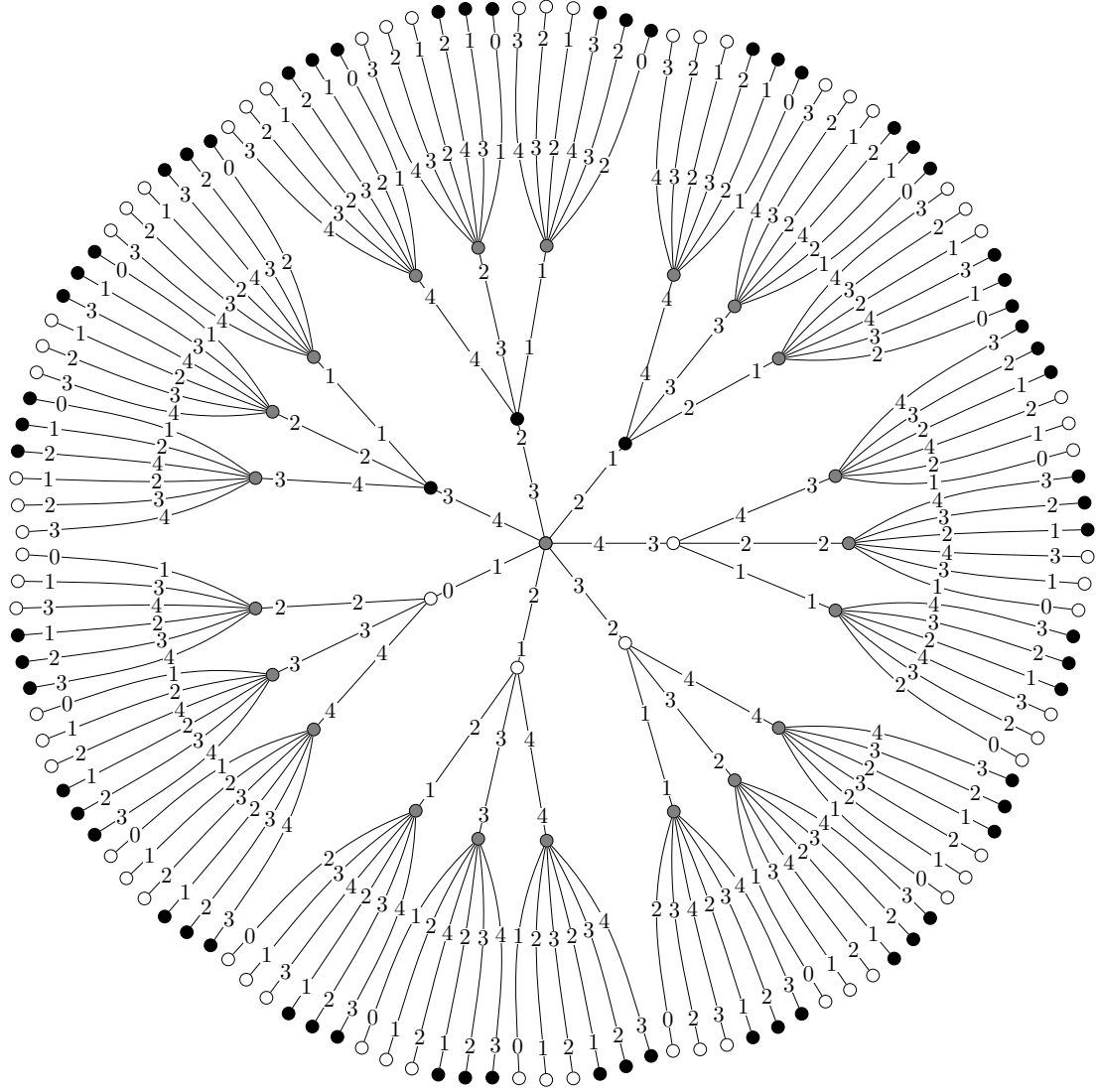


Figure 5: A part of the graph $H_{W,4}$. The node in the centre is node \emptyset . The numbers pictured are outgoing port numbers.

Next, we will define a partial mapping $f_{v,u}: V_C \rightarrow V_C$ for each pair of grey nodes \hat{v} and \hat{u} in \hat{H}_C . Assume that $v = (a_1, a_2, \dots, a_i)$ and $u = (b_1, b_2, \dots, b_j)$. If $v' = (a_1, a_2, \dots, a_i, c_1, c_2, \dots, c_{i'}) \in V_C$ for some $c_1, c_2, \dots, c_{i'}$, and we have

$$f_C((a_1, a_2, \dots, a_i, c_1)) = \bar{C} \quad \text{and} \quad u' = (b_1, b_2, \dots, b_j, c_1, c_2, \dots, c_{i'}) \in V_C,$$

then we define $f_{v,u}(v') = u'$. The idea here is that the subtrees of H_C that have the nodes v and u as their roots and that are not contained in the subgraph \hat{H}_C (except for the root nodes) are isomorphic (up to a certain distance). The mapping $f_{v,u}$ is a partial isomorphism between such subtrees, as one can quite easily check. In what follows, we will use $f_{v,u}$ to show that the $r\text{-}\mathcal{SV}$ -bisimilarity of the nodes $((1, 0, C))$ and $((2, 1, C))$ in \hat{H}_C can be extended to the supergraph H_C .

For each $C \in \{B, W\}$, denote the nodes \emptyset , $((1, 0, C))$ and $((2, 1, C))$ of H_C by v_C , u_C and w_C , respectively. In accordance with our previously introduced notation, denote the corresponding nodes of the subgraph \hat{H}_C by \hat{v}_C , \hat{u}_C and \hat{w}_C .

Lemma 22. *Let $\hat{v}, \hat{u} \in \hat{V}_C$ be grey nodes and let $t \in \mathbb{N}$ be such that $v \leftrightarrow_t^{\mathcal{SV}} u$. If $w \in \text{dom}(f_{v,u})$, $\text{dist}(w, v_C) < 2d - t$ and $\text{dist}(f_{v,u}(w), v_C) < 2d - t$, then $w \leftrightarrow_t^{\mathcal{SV}} f_{v,u}(w)$.*

Proof. We proceed by induction on t . The base case $t = 0$ is straightforward: Since $\text{dist}(w, v_C) < 2d$ and $\text{dist}(f_{v,u}(w), v_C) < 2d$, we have $\deg(w) = \deg(f_{v,u}(w))$. Additionally, observe that we have $f_C(w) = f_C(f_{v,u}(w))$. It follows that we have $w \leftrightarrow_0^{\mathcal{SV}} f_{v,u}(w)$.

For the inductive case, assume that the claim holds for $t = s$ and that $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. If $w = v$, then $f_{v,u}(w) = u$ and we have nothing to prove. Hence, assume $w \neq v$. Denote the neighbours of w by w_1, w_2, \dots, w_k . Then the neighbours of $f_{v,u}(w)$ are $f_{v,u}(w_i)$, $i = 1, 2, \dots, k$. We have $w_i \in \text{dom}(f_{v,u})$ for all i . Additionally, since $\text{dist}(w, v_C) < 2d - (s + 1)$ and $\text{dist}(f_{v,u}(w), v_C) < 2d - (s + 1)$, we have $\text{dist}(w_i, v_C) < 2d - s$ and $\text{dist}(f_{v,u}(w_i), v_C) < 2d - s$ for all i . Now the inductive hypothesis implies that $w \leftrightarrow_s^{\mathcal{SV}} f_{v,u}(w)$ and $w_i \leftrightarrow_s^{\mathcal{SV}} f_{v,u}(w_i)$ for all i . Additionally, it follows immediately from the definition of $f_{v,u}$ that we have $\pi_C(w_i, w) = \pi_C(f_{v,u}(w_i), f_{v,u}(w))$ for all i . Now by Definition 4 we have $w \leftrightarrow_{s+1}^{\mathcal{SV}} f_{v,u}(w)$. Hence the claim holds for $t = s + 1$. \square

Lemma 23. *Let $t \in \mathbb{N}$ and let $\hat{v}, \hat{u} \in \hat{V}_C$ be such that $\text{dist}(\hat{v}, \hat{v}_C) < 2d - t$ and $\text{dist}(\hat{u}, \hat{v}_C) < 2d - t$. If $\hat{v} \leftrightarrow_t^{\mathcal{SV}} \hat{u}$, then $v \leftrightarrow_t^{\mathcal{SV}} u$.*

Proof. We prove the claim by induction on t . The base case $t = 0$ is easy: If $\hat{v} \leftrightarrow_0^{\mathcal{SV}} \hat{u}$, then $f_C(\hat{v}) = f_C(\hat{u})$, and thus $f_C(v) = f_C(u)$. As v and u are of the same colour and neither of them is a leaf node, $\deg(v) = \deg(u)$. Hence $v \leftrightarrow_0^{\mathcal{SV}} u$.

For the inductive step, assume that the claim holds for $t = s$ and that $\hat{v} \leftrightarrow_{s+1}^{\mathcal{SV}} \hat{u}$, where $\text{dist}(\hat{v}, \hat{v}_C) < 2d - (s + 1)$ and $\text{dist}(\hat{u}, \hat{v}_C) < 2d - (s + 1)$. Denote the neighbours of \hat{v} and \hat{u} by $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_d$ and $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_d$, respectively. We have $\hat{v} \leftrightarrow_s^{\mathcal{SV}} \hat{u}$, and by definition, for each \hat{v}_i there is \hat{u}_{j_i} such that $\hat{v}_i \leftrightarrow_s^{\mathcal{SV}} \hat{u}_{j_i}$ and $\pi_C(\hat{v}_i, \hat{v}) = \pi_C(\hat{u}_{j_i}, \hat{u})$, and vice versa. We have $\text{dist}(\hat{v}_i, \hat{v}_C) < 2d - s$ and $\text{dist}(\hat{u}_{j_i}, \hat{v}_C) < 2d - s$ for all i . Now the inductive hypothesis implies that $v \leftrightarrow_s^{\mathcal{SV}} u$, $v_i \leftrightarrow_s^{\mathcal{SV}} u_{j_i}$ for all i and $v_{i_j} \leftrightarrow_s^{\mathcal{SV}} u_j$ for all j .

Since $v \leftrightarrow_s^{\mathcal{SV}} u$, nodes v and u are of the same colour. If they are of colour C , they do not have neighbours other than v_1, v_2, \dots, v_d and u_1, u_2, \dots, u_d , respectively. Then it follows from the definition that $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. Otherwise, v and u are grey, and in addition to v_i and u_i , $i = 1, 2, \dots, d$, they have neighbours generated by rule (H3) or rule (H6). Denote those neighbours by $v'_1, v'_2, \dots, v'_{d-1}$ and $u'_1, u'_2, \dots, u'_{d-1}$, respectively, such that we have $f_{v,u}(v'_i) = u'_i$ for all i . Observe that $\text{dist}(v'_i, v_C) < 2d - s$ and $\text{dist}(u'_i, v_C) < 2d - s$ for all i . Now Lemma 22 shows that $v'_i \leftrightarrow_s^{\mathcal{SV}} u'_i$ for all i . In addition, the definition of $f_{v,u}$ implies that $\pi_C(v'_i, v) = \pi_C(u'_i, u)$ for all i . We have shown that conditions (B2) and (B3) hold also for the additional neighbours, and consequently $v \leftrightarrow_{s+1}^{\mathcal{SV}} u$. Hence the claim is true for $t = s + 1$. \square

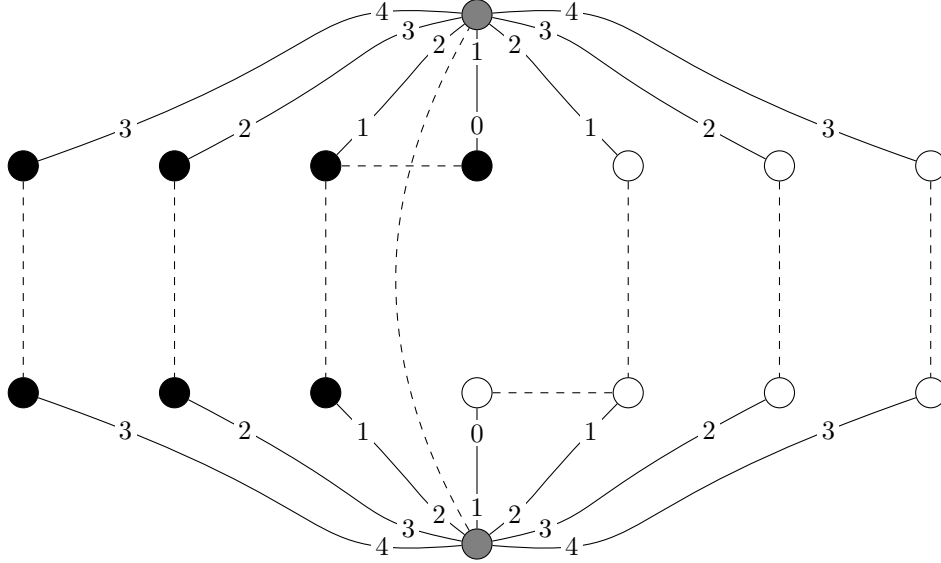


Figure 6: Graphs $H_{B,4}$ and $H_{W,4}$ up to distance one from the root nodes. The dashed lines represent r -SV-bisimilarity between nodes.

Now we can combine our previous results to obtain bisimilarity between certain nodes in the graph H_C for each $C \in \{B, W\}$. Lemma 18 shows that $((1, 0)) \leftrightarrow_{2d-3}^{SV} ((2, 1))$, where $((1, 0))$ and $((2, 1))$ are nodes in the graph G_d . Observe that $g_C(\hat{u}_C) = ((1, 0))$ and $g_C(\hat{w}_C) = ((2, 1))$. Now Lemma 21 implies that $\hat{u}_C \leftrightarrow_{2d-3}^{SV} \hat{w}_C$. We have $\text{dist}(\hat{u}_C, \hat{v}_C) = 1 < 2d - (2d - 3)$ and $\text{dist}(\hat{w}_C, \hat{v}_C) = 1 < 2d - (2d - 3)$. Hence it follows from Lemma 23 that $u_C \leftrightarrow_{2d-3}^{SV} w_C$, where u_C and w_C are neighbours of v_C in the graph H_C .

As in the proof of Theorem 1, we define a port numbering p'_C for each $C \in \{B, W\}$ based on the generalised port numbering p_C . Again, we need to preserve bisimilarity as well as have identical outgoing port numbers from nodes u_C and w_C towards node v_C . Define function f from the set of all generalised ports of H_C to $[2d - 1]$ as follows: $f(1, B) = f(1, W) = 1$, $f(i, B) = 2i - 1$ and $f(i, W) = 2i - 2$ for all $i = 2, 3, \dots, d$, $f(0, G) = 1$ and $f(i, G) = i$ for all $i = 1, 2, \dots, d$. Then, if $p_C(v, a) = (u, b)$ for some nodes v, u and port numbers a, b , set $p'_C(v, f(a)) = (u, f(b))$. Without too much effort, one can check that p'_C is indeed a valid port numbering of H_C , and that we have $\pi'_C(u_C, v_C) = 1 = \pi'_C(w_C, v_C)$. Lemma 7 implies that $(H_C, f_C, u_C, p'_C) \leftrightarrow_{2d-3}^{SV} (H_C, f_C, w_C, p'_C)$.

To reach our ultimate goal, we need to define one more mapping. Define $h: V_B \rightarrow V_W$ as follows: if $v = (a_1, a_2, \dots, a_i) \in V_B$, where $i \geq 1$ and $a_1 = (b_1, b_2, C)$ for some $b_1 \geq 2$, set $h(v) = u$, where $u = (a_1, a_2, \dots, a_i) \in V_W$. Additionally, set $h(v_B) = v_W$. Thus, there is one subtree starting from a child of v_B , the one having the node $u_B = ((1, 0, B))$ as its root, that is excluded from the domain of h . Similarly, the subtree having $u_W = ((1, 0, W))$ as its root is excluded from the range of h . See Figure 6 for an illustration of the situation.

Lemma 24. *Let $v \in V_B$ and $u \in V_W$ be nodes such that $h(v) = u$. Then for all $t = 0, 1, \dots, 2d - 2$ we have $(H_B, f_B, v, p'_B) \leftrightarrow_t^{SV} (H_W, f_W, u, p'_W)$.*

Proof. We prove the claim by induction on t . The base case $t = 0$ is trivial: if $h(v) = u$, then by definition of h we have $\deg(v) = \deg(u)$ and $f_B(v) = f_W(u)$ and therefore $v \leftrightarrow_0^{SV} u$.

For the inductive step, suppose that the claim holds for $t = s < 2d - 2$. Consider two arbitrary nodes $v \in V_B$ and $u \in V_W$ such that $h(v) = u$. By the inductive hypothesis we have $v \leftrightarrow_s^{SV} u$. If $v \neq v_B$, all the neighbours of v are in the domain of h and all the neighbours of u are in the range of h . Furthermore, if w is a neighbour of v , we have $\pi'_B(w, v) = \pi'_W(h(w), u)$, and by the inductive hypothesis, $w \leftrightarrow_s^{SV} h(w)$. Now Definition 4 implies that $v \leftrightarrow_{s+1}^{SV} u$.

If $v = v_B$, v has one neighbour that is not in $\text{dom}(h)$. That neighbour is $u_B = ((1, 0, B))$. Similarly, $h(v) = v_W$ has one neighbour that is not in the range of h , namely $u_W = ((1, 0, W))$. However, as shown above, we have $u_B \leftrightarrow_{2d-3}^{SV} w_B$, and thus $u_B \leftrightarrow_s^{SV} w_B$. Since we have also $w_B \leftrightarrow_s^{SV} h(w_B)$, Lemma 6 implies that $u_B \leftrightarrow_s^{SV} h(w_B)$. Additionally, we have

$$\pi'_B(u_B, v) = \pi'_B(w_B, v) = \pi'_B(h(w_B), u).$$

Similarly, we have $u_W \leftrightarrow_s^{SV} w_W$ and $w_W \leftrightarrow_s^{SV} h^{-1}(w_W)$, from which we get $u_W \leftrightarrow_s^{SV} h^{-1}(w_W)$. Additionally,

$$\pi'_W(u_W, u) = \pi'_W(w_W, u) = \pi'_W(h^{-1}(w_W), v).$$

We have shown that conditions (B1)–(B3) hold even if considering also neighbours not handled by the mapping h , and consequently we have $v \leftrightarrow_{s+1}^{SV} u$. Thus the claim holds for $t = s + 1$. \square

Let $d \geq 2$ and $\Delta = 2d - 1$. Then $H_{B,d}, H_{W,d} \in \mathcal{F}(\Delta)$. Let $\mathcal{A} \in \mathcal{SV}$ be any algorithm with a running time at most $\Delta - 1 = 2d - 2$. Consider the execution of \mathcal{A} in the nodes $v_B \in V_{B,d}$ and $v_W \in V_{W,d}$. Now Lemma 24 together with Lemma 5 implies that \mathcal{A} produces the same output in v_B and v_W . Recall that for any valid solutions $S \in \Pi(H_{B,d}, f_B)$ and $S' \in \Pi(H_{W,d}, f_W)$ we have $S(v_B) \neq S'(v_W)$. Hence \mathcal{A} does not solve the problem Π . This concludes the proof of Theorem 2.

Remark 25. Note that we could define a similar problem without local inputs, by encoding the colours in the structure of the graph. One way to do this is to add one new neighbour to each black node and two new neighbours to each white node. If $d \geq 3$, this does not increase the maximum degree of the graph. Then we could define the set of solutions to consist of, for example, mappings S such that $S(v) = 1$ if node v has an odd number of neighbours of an odd degree and $S(v) = 0$ otherwise. However, for illustrative purposes, it was beneficial the use a colouring instead.

5 Conclusions

To sum up, we have shown that the simulation technique used to prove $SV = MV$ is optimal in the following sense: breaking symmetry between incoming messages is always possible in time $2\Delta - 1$, and there are graphs where $2\Delta - 1$ rounds are strictly required. Furthermore, we have constructed a graph problem for which the difference in running time between algorithms in SV and MV is linear in Δ . This settles the last significant open question related to the hierarchy studied by Hella et al. [8].

Acknowledgements

This manuscript is based on the author's master's thesis [10] submitted to the Department of Mathematics and Statistics of the University of Helsinki. The author would like to thank Jukka Suomela for guidance and feedback as well as Lauri Hella and Juha Kontinen for comments on the thesis. A minor part of the research was conducted while the author was employed at the Department of Computer Science of the University of Helsinki.

References

- [1] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, Laszlo Lovász and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. 20th Annual Symposium on Foundations of Computer Science (FOCS 1979)*, pages 218–223. IEEE, 1979. DOI: 10.1109/SFCS.1979.34.
- [2] Patrick Blackburn, Johan van Benthem and Frank Wolter, editors. *Handbook of Modal Logic*. Studies in Logic and Practical Reasoning 3. Elsevier, Amsterdam, 2007.

- [3] Patrick Blackburn, Maarten de Rijke and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science 53. Cambridge University Press, Cambridge, UK, 2001.
- [4] Paolo Boldi, Shella Shammah, Sebastiano Vigna, Bruno Codenotti, Peter Gemmell and Janos Simon. Symmetry breaking in anonymous networks: characterizations. In *Proc. 4th Israel Symposium on the Theory of Computing and Systems (ISTCS 1996)*, pages 16–26. IEEE, 1996.
- [5] Paolo Boldi and Sebastiano Vigna. Computing vector functions on anonymous networks. In *Proc. 4th Colloquium on Structural Information and Communication Complexity (SI-ROCCO 1997)*, pages 201–214. Carleton Scientific, 1997.
- [6] Jérémie Chalopin. Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes. PhD thesis. LaBRI, Université Bordeaux 1, 2006.
- [7] Yuval Emek and Roger Wattenhofer. Stone age distributed computing. In *Proc. 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC 2013)*, pages 137–146. ACM Press, 2013. DOI: 10.1145/2484239.2484244. arXiv: 1202.1186.
- [8] Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempinen, Kerkko Luosto, Jukka Suomela and Jonni Virtama. Weak models of distributed computing, with connections to modal logic. In *Distributed Computing* 28.1 (2015), pages 31–53. DOI: 10.1007/s00446-013-0202-3. arXiv: 1205.2051.
- [9] Michal Koucký. Universal traversal sequences with backtracking. In *Journal of Computer and System Sciences* 65.4 (2002), pages 717–726. DOI: 10.1016/S0022-0000(02)00023-5.
- [10] Tuomo Lempinen. A Classification of Weak Models of Distributed Computing. MSc thesis. University of Helsinki, Department of Mathematics and Statistics, 2014. HDL: 10138/144214.
- [11] Masafumi Yamashita and Tsunehiko Kameda. Leader election problem on networks in which processor identity numbers are not distinct. In *IEEE Transactions on Parallel and Distributed Systems* 10.9 (1999), pages 878–887. DOI: 10.1109/71.798313.